МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра Прикладная математика и информатика
(наименование)
09.04.03 Прикладная информатика
(код и наименование направления подготовки)
Управление корпоративными информационными процессами
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему «Исследование технологии интеграции Amazon Web Services (AWS) для обеспечения высокой доступности и масштабируемости веб-приложения»

Обучающийся И. М. Приказчиков
(Инициалы Фамилия) (личная подпись)

Научный руководитель к.п.н., доцент, О.В. Оськина
((ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия))

Тольятти 2024

Оглавление

Введение

В современном цифровом мире, где электронная коммерция играет важную роль в международной торговле, успешное функционирование интернет-магазинов и маркетплейсов становится ключевым фактором для достижения конкурентных преимуществ и удовлетворения потребностей клиентов. Неспособность организации обеспечить бесперебойный доступ к возможности покупки своей продукции ведет не только к снижению продаж, но и к репутационным рискам. Компании, особенно те, которые нацелены на расширение зоны своей работы, сталкиваются с большим количеством трудностей при использовании традиционных подходов управления инфраструктурой. В рамках данной работы мы взаимодействовали с ООО «Тотал Продакшн», одним из крупнейших продавцов и поставщиков концертного оборудования в Самарской области. Руководство компании осознает важность глобализации и планирует расширить свою деятельность, выпустив свой интернет-магазин на международный рынок.

Актуальность данного исследования обусловлена проблемой обеспечения эффективного масштабирования к нагрузке и непрерывной доступности веб-приложений для пользователей со всего мира. Данная проблема является актуальной и востребованной в сфере информационных технологий в связи с ростом объема данных, пользовательской нагрузки и требований к системам, что ставит перед инженерами задачу в умелом проектировании новых систем и поискам новых решений. Исследование этой проблемы не только поможет компании ООО «Тотал Продакшн» решить свои текущие задачи, но и обеспечит вклад в систематизацию новых знаний и практических рекомендаций в области оптимизации инфраструктуры веб-приложений, которые могут быть применены в других проектах.

Проблема исследования. Современные веб-приложения требуют обеспечения высокой доступности и масштабируемости, особенно в условиях выхода на международный рынок. Однако традиционные локальные серверные

инфраструктуры не способны удовлетворить эти требования из-за ограниченной пропускной способности, сложности в управлении и высокой стоимости модернизации. Это ставит перед научным сообществом и IT-индустрией задачу разработки и внедрения новых подходов к использованию облачных технологий, которые могли бы обеспечить гибкость, отказоустойчивость и экономическую эффективность при проектировании и эксплуатации веб-приложений.

Актуальность проблемы обусловлена не только фундаментальными изменениями цифровых систем, но и релаьными ситуациями, с которыми сталкиваются предприятия по всему миру. Например, существующая локальная инфраструктура для работы интернет-магазина ООО «Тотал Продакшн» является препятствием на пути международной эскпансии бизнеса ввиду своих технических ограничений. Поэтому, для решения поставленной задачи необходимо изучить методы миграции и оптимизации архитектуры веб-приложений, учитывая актуальные требования к нагрузке и доступности, предъявляемые к приложениям в сфере электронной коммерции.

Объект исследования – процесс проектирования и оптимизации архитектуры веб-приложений для обеспечения их высокой доступности и масштабируемости в условиях динамично изменяющихся пользовательских нагрузок и технических ограничений.

Предмет исследования – автоматизация процесса управления архитектурой веб-приложения с использованием облачных технологий, включая предиктивное масштабирование, оптимизацию ресурсов и управление отказоустойчивостью.

Гипотеза исследования: если для проектирования системы интернет-магазина будут применены облачные технологии, то система сможет эффективно масштабироваться в ответ на изменяющийся интернет-трафик, обеспечивая высокую доступность и отказоустойчивость, минимизируя простои и перерывы в обслуживании, а также сохранять оптимальную производительность при изменении нагрузки.

Цель данной научной работы: исследовать и разработать качественно новый подход к интеграции облачных сервисов для обеспечения высокой доступности и масштабируемости веб-приложения направления «электронная коммерция», учитывающий современные технические и экономические ограничения. В рамках данной научной работы основной фокус направлен на создание архитектуры веб-приложения, способной эффективно справляться с возростающими требованиями к производительности и устойчивости. Для достижения этой цели были сформулированы следующие задачи:

- провести анализ существующих подходов к размещению инфраструктуры веб-приложения для обеспечения высокой доступности и масштабируемости, включая локальные и облачные инфраструктуры;
- провести сравнение функционала для построения масштабируемой и отказоустойчивой системы и выбрать провайдера облачных сервисов, который будет наиболее эффективным в контексте построения веб-приложения в сфере электронной коммерции;
- разработать собственное уникальное решение, включающее архитектуру веб-приложения на основе выбранного облачного провайдера, которое учитывает технические ограничения и бюджетные требования, включая возможности масштабирования, отказоустойчивости и высокодоступности;
- провести апробацию разработанного решения на примере интернет-магазина компании ООО «Тотал Продакшн» для проверки его эффективности, подтверждения гипотезы исследования и оценки соответствия критериям высокой доступности и масштабируемости.

Для выполнения задач исследования были использованы следующие методы:

- анализ и сравнение видов инфраструктур и провайдеров предоставляющих облачные услуги в контексте мастшабируемости, высокой досупности, экономической эффективности и безопасности;

- моделирование и проектирование облачной архитектуры с помощью инструментов предоставляемых провайдером облачных услуг;
- экспериментальное тестирование новой системы с целью оценки ее производительности и надежности;
- мониторинг и анализ полученных данных.

Теоретическая значимость данного исследования заключается в систематизации накопленного научного материала по избранной теме, а именно по разработке эффективных подходов к масштабированию и обеспечению непрерывного функционирования веб-приложений в сфере электронной коммерции и информационных технологий. Материалы магистерской диссертации вносят вклад в изучение теоретических основ использования облачных технологий в сфере интернет-коммерции в эпоху глобализации и цифровизации.

Новизна исследования. Несмотря на то, что данное исследование имеет связь с предыдущими научно-исследовательскими работами, посвященными проблемам эффективного функционирования веб-приложений и их инфраструктуре, результаты этого исследования будут расширять существующие знания и опыт в данной области, а также предоставят практические рекомендации и качественно новое архитектурное решение, обеспечивающее масштабируемость и отказоустойчивость веб-приложений. Таким образом, оно может быть применено не только в контексте интернет-магазина ООО «Тотал Продакшн», но и в других проектах отрасли электронной коммерции.

Глава 1 Современные подходы к обеспечению высокой доступности и масштабируемости веб-приложений

В настоящее время веб-приложения представляют собой сложные системы, которые должны обеспечивать производительность, надежность и доступность в условиях постоянно растущего числа пользователей и объема данных. Простой на небольшие промежутки времени может привести к значительным финансовым потерям, снижению лояльности клиентов и ухудшению репутации компании. Масштабируемость, в свою очередь, позволяет интернет-магазину адаптироваться к изменениям нагрузки, особенно в пиковые периоды, такие как праздничные распродажи или акции. [16]

В целом, высокая доступность и масштабируемость являются фундаментальными аспектами проектирования современных веб-приложений. В условиях увеличения нагрузки на интернет-ресурсы разработчики сталкиваются с необходимостью создавать системы, обеспечивающие минимальные временные задержки и гарантированное предоставление услуг конечным пользователям. [12] Согласно современным исследованиям, ключевыми характеристиками высокодоступных систем являются устойчивость к отказам, автоматическое восстановление и непрерывная доступность.

Высокая доступность предполагает, что система способна функционировать без простоев в течение заранее определённого времени. Наиболее часто применяемым критерием оценки доступности является показатель SLA (Service Level Agreement), который выражается в процентах времени бесперебойной работы. Например, показатель «пять девяток» (99.999%) соответствует простою не более 5 минут в год.

Большинство поставщиков услуг связи в режиме реального времени соответствуют уровням обслуживания, которые обеспечивают доступность от 99,9% до 99,999%. В зависимости от желаемой степени высокой доступности необходимо принимать все более сложные меры на протяжении всего жизненного цикла приложения.

Необходимо проектировать системы так, чтобы не было единой точки отказа. Важно использовать автоматизированные механизмы мониторинга, обнаружения отказов и отказоустойчивости как для компонентов без состояния, так и для компонентов с состоянием.

Единичные точки отказа (SPOF) обычно устраняются с помощью конфигурации избыточности N+1 или 2N, где N+1 достигается за счет балансировки нагрузки между узлами «активный-активный», а 2N достигается парой узлов в конфигурации «активный-резервный». [2]

Системы высокой доступности проектируются с использованием нескольких архитектурных принципов:

- избыточность – создание резервных компонентов предотвращает полное отключение в случае отказа основного узла, как пример – резервное копирование серверов и использование отказоустойчивых баз данных;
- балансировка нагрузки – методы равномерного распределения пользовательских запросов между серверами минимизируют вероятность перегрузок;
- мониторинг и автоматическое восстановление – использование систем мониторинга, позволяет отслеживать работоспособность приложений и запускать механизмы самовосстановления в случае инцидентов.

1.1 Сравнение локальной инфраструктуры и облачных технологий для обеспечения высокой доступности и масштабируемости веб-приложений

Традиционно для размещения веб-приложений компании используют локальную инфраструктуру, которая предполагает, что серверы и всё оборудование размещаются и обслуживаются внутри компании. Такой подход обеспечивает полный контроль над системой, что делает локальную инфраструктуру привлекательной для бизнеса с повышенными требованиями к

безопасности данных. [22] Компании могут настраивать оборудование под свои специфические задачи, достигая максимальной производительности.

Для обеспечения высокой доступности и масштабируемости организации зачастую прибегают к традиционному методу решения этих задач – расширению локальной инфраструктуры путем приобретения новых комплектующих для повышения мощностей серверов, и, как следствие, нередко сталкиваются с ограничениями, которые данный подход подразумевает.

Несмотря на неоспоримые преимущества, недостатки очевидны – локальная инфраструктура требует значительных вложений со стороны владельцев бизнеса, а именно: расходы на покупку серверов, систем хранения данных, маршрутизаторов и резервных источников питания. Помимо этого, масштабирование локальной инфраструктуры является сложным и времязатратным процессом, так как добавление новых серверов требует места, ресурсов и времени. Все это делает оперативное реагирование на внезапный рост нагрузки затруднительным, учитывая всю непредсказуемость процессов. [17]

Локальная инфраструктура также требует наличия квалифицированного ИТ-персонала для обслуживания серверов, их модернизации и обеспечения безопасности, что создает дополнительную финансовую нагрузку. Кроме этого, локальная инфраструктура при непредсказуемой нагрузке гарантирует высокий риск отказов – оборудование, вышедшее из строя из-за слишком большой нагрузки зачастую приводит к простою системы. Соответственно, для минимизации таких рисков необходимо дублировать ресурсы, что, в свою очередь, еще больше увеличивает затраты владельцев.

Наконец, локальная инфраструктура ограничена в возможностях географического распределения. Развертывание серверов в разных регионах снова сопряжено с огромными затратами, которые могут быть недоступны для большинства компаний. Таким образом, несмотря на полную автономию и контроль, локальная инфраструктура имеет множество существенных ограничений, которые необходимо учитывать при ее использовании.

Облачные технологии, или же cloud-computing, противопоставляются традиционному подходу с локалькой инфраструктурой. Облачные вычисления появились как новая технология в конце 2010-х, и с каждым годом их актуальность и популярность только возрастает. Облако можно рассматривать как концептуальный слой в Интернете, который предоставляет доступ к имеющимся программным и аппаратным ресурсам через визуальный интерфейс. Ключевыми компонентами его нынешней популярности стали такие концепции, как самообслуживание по требованию, широкий сетевой доступ и объединение ресурсов. Облачные вычисления стали особенно привлекательными среди современных пользователей и владельцев бизнеса, ведь они не только минимизируют вложения в инфраструктуру и затраты на управление ресурсами, но и представляют при этом довольно гибкий спектр услуг. [26]

Облачные платформы, такие как Amazon Web Services (AWS), Microsoft Azure и Google Cloud Platform (GCP), а также их отечественные аналоги как Yandex Cloud, VK Cloud играют центральную роль в разработке систем, способных адаптироваться к изменяющимся требованиям бизнеса. Гибкость, автоматизация управления ресурсами и экономическая эффективность делают их привлекательными как для крупных корпораций, так и для малых предприятий. [1]

Помимо использования облачных технологий обособленно, проблему доступности и масштабируемости можно решать через использование гибридных облаков (сочетания локальных серверов с облачными ресурсами) или через мультиоблачные архитектуры (развертывание приложений в нескольких облаках).

Одним из главных достоинств облачных технологий по сравнению с локальной инфраструктурой является их масштабируемость. Облачные платформы позволяют быстро увеличивать или уменьшать ресурсы в зависимости от текущих потребностей бизнеса. Это особенно важно для интернет-магазинов, которые обычно подвергаются пиковым нагрузкам в определенные моменты (такие как праздничные распродажи или акции). [28]

Высокая доступность является еще одним важным преимуществом облачных решений. Провайдеры облачных услуг, предлагают соглашения об уровне обслуживания (SLA), которые гарантируют 99.9% доступности и выше. Это достигается за счет использования нескольких дата-центров, распределённых по нескольким регионам по всему миру, что обеспечивает отказоустойчивость и минимизирует риски простоев. В случае сбоя одного из дата-центров, данные и приложения мгновенно перенаправляются на другие ресурсы, обеспечивая непрерывность бизнес-процессов.

Снижение затрат – еще один важный аспект, обеспечивающий привлекательность облачных технологий для владельцев бизнеса. Облачные решения устраняют необходимость в больших вложениях в покупку и обслуживание серверов, систем хранения данных и другого оборудования. Вместо этого компании оплачивают только те ресурсы, которые они фактически используют, что делает финансовые расходы более гибкими и предсказуемыми. Более того, нет необходимости иметь в штате специалистов для обслуживания серверов, что также сокращает затраты на персонал. [14]

Еще одним преимуществом является удобство управления. Современные облачные платформы предоставляют пользователям мощные инструменты для мониторинга, автоматизации и управления ресурсами, что значительно упрощает процесс управления и минимизирует риски ошибок, связанных с человеческим фактором.

Также стоит отметить простоту резервного копирования и восстановления данных. В облаке данные часто автоматически сохраняются в нескольких местах, что обеспечивает их защиту от потерь и позволяет быстро восстанавливать информацию в случае сбоя. [24]

Тем не менее, несмотря на преимущества, облачные решения имеют и несколько недостатков, главным из которых является зависимость от конкретного провайдера. Когда компания решает перейти на облачные технологии, она становится зависимой от политики и ценовой стратегии

выбранного облачного провайдера, что может вызвать проблемы в случае изменений тарифов, условий обслуживания или политики безопасности.

Требования к интернет-соединению также могут стать ограничением для некоторых организаций. Для нормальной работы облачных сервисов требуется стабильное и быстрое интернет-соединение. Если связь с интернетом прерывается или становится ненадежной, это может привести к недоступности важных сервисов и данных.

Тем не менее, несмотря на перечисленные недостатки, ощутимые преимущества и новые возможности облачных технологий делают их неотъемлемой частью современного программного обеспечения. В свою очередь, разработчики получают доступ к инструментам, помогающим в создании конкурентоспособных, надежных и производительных приложений.

1.2 Сравнение возможностей облачных провайдеров для обеспечения высокой доступности и масштабируемости веб-приложений

Разнообразие и многогранность облачных ресурсов требуют не только тщательного изучения и последующего тщательного планирования архитектуры, но и ставят перед разработчиками высокую планку при проектировании облачных систем. Основные вызовы при проектировании современных веб-приложений включают:

- сложность архитектуры, т.к. большинство приложений состоят из множества взаимосвязанных компонентов (серверы приложений, базы данных, API, и систем мониторинга), и каждый из этих компонентов должен быть оптимизирован для максимальной производительности;
- управление нагрузкой, особенно в периоды пикового спроса (во время сезонных распродаж или глобальных событий), ведь приложения должны обрабатывать резко возрастающий объем запросов без снижения скорости работы;

- обеспечение безопасности и защиты данных пользователей, которые являются конфиденциальной информацией и должны быть защищены на самом высоком уровне.

Облачные технологии способны удовлетворить самые высокие требования и включают в себя виртуализацию, контейнеризацию, автоматизированное масштабирование и отказоустойчивость. Все это позволяет разработчикам сосредоточиться на улучшении пользовательского опыта, а не на управлении инфраструктурой. [34]

Существуют различные провайдеры облачных услуг, предлагающие решения для масштабируемости и доступности. Сравним 5 самых популярных из них, включив в перечень как российских, так и зарубежных провайдеров для объективности исследования.

1. Microsoft Azure.

   Сервис предоставляет мощные инструменты для работы с корпоративными системами, включая интеграцию с Windows Server и Active Directory.

   Azure предлагает масштабируемость через Azure Auto-scaling, который автоматически регулирует количество виртуальных машин в зависимости от нагрузки. Также Azure предоставляет Azure Load Balancer, поддерживающий как глобальное, так и локальное распределение нагрузки, что увеличивает гибкость масштабирования и высокую доступность через региональные репликации данных. [13]

   Провайдер предлагает 99.99% SLA и поддерживает несколько регионов и Availability Zones для развертывания приложений с высокой доступностью. Механизмы Azure Availability Sets и Azure Zones позволяют повысить отказоустойчивость.

   Цена также гибкая, с оплатой по использованию, и часто бывает привлекательной для компаний, уже использующих Microsoft-продукты (например, Windows Server, SQL Server и т.д.). Azure предоставляет различные варианты скидок и контрактов.

Anure предоставляет Azure Monitor, который включает мониторинг инфраструктуры, приложений, а также безопасность и производительность. [13]

Microsoft Azure сильна в области безопасности благодаря интеграции с Active Directory, а также предлагает решения для шифрования и соблюдения стандартов безопасности.

Microsoft Azure является надежным выбором для компаний, особенно тех, кто уже использует экосистему продуктов Microsoft. Интеграция с корпоративными системами, гибкость масштабирования и высокая доступность делают Azure подходящим решением для построения устойчивых и производительных приложений. Широкий набор инструментов мониторинга, продвинутые функции безопасности и конкурентоспособная ценовая политика обеспечивают высокую востребованность Azure как для крупных корпораций, так и для организаций среднего бизнеса, стремящихся к облачной трансформации.

2. Amazon Web Services (AWS)

Данный облачный провайдер является лидером на рынке облачных услуг благодаря широкой экосистеме инструментов и глобальной инфраструктуре.

AWS обладает мощными инструментами для масштабирования, такими как Auto Scaling, который позволяет автоматически добавлять или удалять ресурсы в зависимости от нагрузки. Кроме того, сервисы AWS интегрируются с Elastic Load Balancer (ELB) для распределения нагрузки между серверами, что способствует балансировке нагрузки и предотвращению перегрузки. Масштабируемость происходит быстро и гибко. [3]

Amazon Web Services обеспечивает высокую доступность и предлагает 99.99% SLA, этого можно достичь при промощи размещения ресурсов в нескольких географически удаленных регионах и Availability Zones. Глобальная сеть Amazon включает в себя более 100 зон доступности

более чем в 30 регионах, что обеспечивает повышенную отказоустойчивость. Благодаря AWS инженеры могут развертывать приложения через Multi-AZ (мульти-зональные) и Multi-Region решения, это позволяет дополнительно повысить доступность приложения. Сервисы, такие как Relational Database Service (RDS) и S3, предоставляют механизмы для обеспечения резервирования данных.

AWS отличается гибкой ценовой политикой: оплата по использованию, скидки при долгосрочных резервированиях через Reserved Instances и Savings Plans, а также вариант Spot Instances для экономии на нерегулярных задачах. Это делает AWS экономически эффективным выбором как для стартапов, так и для крупных корпораций.

Немаловажным является факт того, что провайдер предлагает обширные инструменты для мониторинга, такие как CloudWatch, который позволяет отслеживать состояние всех компонентов инфраструктуры и устранять проблемы в реальном времени. [8]

Вопросы безопасности решаются через AWS Identity and Access Management (IAM), шифрование данных (KMS), а также соответствие стандартам, таким как GDPR, HIPAA и PCI DSS. Одной из сильных сторон AWS является огромное количество сервисов и интеграций, включая серверлесс-вычисления (AWS Lambda), контейнеры (ECS, EKS), а также инструменты для машинного обучения (SageMaker).

Таким образом, Amazon Web Services является универсальным и мощным решением для построения масштабируемых, высокодоступных и отказоустойчивых приложений. Гибкость ценообразования, широкий спектр сервисов и высокая надежность делают AWS привлекательным как для малых компаний, так и для крупных предприятий. Возможности AWS в области серверлесс-архитектуры, машинного обучения и глобального развертывания позволяют реализовать сложные проекты с минимальными затратами времени и ресурсов, что подтверждает его лидерство на рынке облачных технологий.

3. Google Cloud Platform (GCP)

Данный облачный провайдер известен своей мощной инфраструктурой и инновационными решениями, унаследованными от опыта Google в работе с высоконагруженными системами.

В Google Cloud также есть инструменты для автоматического масштабирования, такие как Google Compute Engine с AutoScaler, который автоматически регулирует количество виртуальных машин на основе метрик, таких как загрузка CPU или запросы к приложениям. Для распределения нагрузки GCP использует Cloud Load Balancing, который поддерживает глобальное распределение трафика, обеспечивая низкую задержку и высокую производительность. [3]

Высокая доступность достигается благодаря глобальной сети центров обработки данных, разделенных на регионы и зоны доступности (Availability Zones). GCP предоставляет инструменты репликации данных, такие как Cloud Spanner и Multi-Regional Buckets в Cloud Storage, которые автоматически синхронизируют данные между регионами. GCP гарантирует SLA до 99.95% доступности для ключевых сервисов.

Модели ценообразования GCP гибкие, с оплатой по факту использования. GCP также предлагает преимущества, такие как автоматические скидки на долгосрочное использование (Sustained Use Discounts) и Preemptible VMs для экономии на нерегулярных вычислительных задачах. Но всё же цена на услуги сервиса привлекательна для тех, кто использует Big Data и Machine Learning сервисы. GCP также предлагает скидки за длительное использование и скидки для стартапов.

В качестве инструментов для мониторинга и логирования в GCP доступен Operations Suite (Stackdriver), который обеспечивает мониторинг, логирование и диагностику в приложениях. Система хорошо интегрирована с другими сервисами GCP.

В области безопасности GCP предлагает Cloud IAM для управления доступом, шифрование данных по умолчанию и соответствие мировым стандартам, включая GDPR и ISO/IEC 27001.

Отличительным преимуществом GCP перед другими облачными провайдерами являются наличие мощных инструменов для работы с данными и машинным обучением. Среди них – BigQuery для анализа больших данных, Vertex AI для разработки моделей машинного обучения и Dataflow для потоковой обработки данных. [5]

В целом, Google Cloud Platform является отличным выбором для компаний, работающих с большими данными, аналитикой и машинным обучением. Благодаря мощной инфраструктуре, высокому уровню безопасности и гибкому ценообразованию, GCP предоставляет решения, которые подходят как для стартапов, так и для крупных корпораций. Инновационные инструменты и интеграция с другими сервисами Google делают GCP сильным конкурентом на рынке облачных технологий, особенно для проектов, ориентированных на высокую производительность и интеллектуальную обработку данных.

4. VK Cloud

VK Cloud (ранее Mail.ru Cloud Solutions) – российская облачная платформа, ориентированная на корпоративные и государственные организации, а также стартапы. Платформа предлагает масштабируемость через автоматическое изменение ресурсов на базе VK Cloud Scaling, позволяя быстро адаптироваться к изменению нагрузки. Для распределения трафика используется Cloud Load Balancer, который поддерживает балансировку как на уровне приложений, так и на сетевом уровне.

Высокая доступность достигается благодаря географически распределённым дата-центрам в России. VK Cloud поддерживает развертывание приложений в нескольких зонах доступности (Availability Zones), что обеспечивает отказоустойчивость и непрерывность работы.

Репликация данных между зонами доступности и резервное копирование через Object Storage и Backup Services увеличивают надежность. Облачный провайдер VK документально подтверждает SLA до 99.95% доступности для своих основных сервисов.

Цены VK Cloud гибкие и конкурентоспособные, с оплатой по использованию и возможностью скидок при долгосрочных контрактах. Платформа предоставляет предсказуемое ценообразование, что особенно важно для организаций, планирующих бюджет.

Для мониторинга и управления инфраструктурой VK Cloud предлагает Monitoring & Alerts, который отслеживает метрики виртуальных машин, сетей и приложений. В области безопасности платформа поддерживает шифрование данных, управление доступом через Cloud IAM и соответствие требованиям российского законодательства, включая ФЗ-152 (о персональных данных).

Особенностью VK Cloud является фокус на локальных клиентах, включая интеграции с российскими корпоративными системами. Платформа предоставляет решения для Big Data через Data Platform и поддержку контейнеризации с помощью Kubernetes (VK Cloud Managed Kubernetes). Это делает VK Cloud удобным выбором для организаций, работающих в пределах России.

5. Yandex Cloud

Ведущая российская облачная платформа, предоставляющая гибкие и масштабируемые решения для бизнеса. Платформа поддерживает автоматическое масштабирование через Managed Instance Groups, позволяя адаптировать инфраструктуру под изменение нагрузки. Для распределения трафика используется Yandex Application Load Balancer, обеспечивающий балансировку как на уровне приложений, так и на сетевом уровне, с поддержкой HTTPS. [31]

Высокая доступность достигается благодаря географически распределённым зонам доступности (Availability Zones) в России. Yandex

Cloud поддерживает репликацию данных и резервное копирование через Managed Databases и Object Storage, что повышает отказоустойчивость. SLA Yandex Cloud гарантирует до 99.99% доступности для основных сервисов, что делает её надёжным выбором для критически важных приложений.

Цены Yandex Cloud прозрачны и конкурентоспособны, с оплатой по факту использования. Платформа предлагает скидки при долгосрочных контрактах, а также выгодные тарифы для стартапов и малого бизнеса.

Для мониторинга и управления Yandex Cloud предоставляет Yandex Monitoring и Yandex Logging, которые позволяют отслеживать метрики приложений, инфраструктуры и сетей. В области безопасности платформа предлагает особый функционал Cloud IAM для управления доступом, шифрования данных, а также полного соответствия российским стандартам, включая ФЗ-152 и требования ФСТЭК.

Одной из сильных сторон Yandex Cloud является интеграция с экосистемой Yandex. Платформа предоставляет мощные инструменты для анализа данных, включая Yandex DataLens и Yandex Managed Service for ClickHouse – один из самых быстрых аналитических движков. Кроме того, Yandex Cloud поддерживает контейнеризацию через Managed Kubernetes, что упрощает разработку и развертывание приложений.

Результаты сравнения облачных провайдеров представлены в таблице 1.

Таблица 1 – Сравнение самых популярных провайдеров облачных услуг в контексте мастшабируемости, высокой досупности, экономической эффективности и безопасности

Платформа	Масштабируемость	Высокая доступность	Экономическая эффективность	Безопасность
Microsoft Azure	★★★★★	★★★★★	★★★☆☆	★★★★★
	Полная поддержка авто-масштабирования и распределения нагрузки с высоким уровнем гибкости.	Глобальная сеть с гарантией 99.99% SLA, Multi-AZ, Multi-Region решения.	Скидки для компаний, уже использующих продукты Microsoft, плата за фактическое использование, скидки для долгосрочных контрактов. Стоимость в среднем выше чем при использовании российских аналогов.	Соответствие международным стандартам GDPR, HIPAA, PCI DSS. Шифрование данных, защита от утечек, управление доступом.
Amazon Web Services (AWS)	★★★★★	★★★★★	★★★★☆	★★★★★
	Полная поддержка авто-масштабирования и распределения нагрузки с высоким уровнем гибкости.	Глобальная сеть с гарантией 99.99% SLA, Multi-AZ, Multi-Region решения.	Гибкие схемы ценообразования, такие как оплата по использованию, скидки на долгосрочные контракты (Reserved Instances и Savings Plans), а также Spot Instances для нерегулярных задач. Стоимость в среднем выше чем при использовании российских аналогов.	Соответствие международным стандартам GDPR, HIPAA, PCI DSS Шифрование данных, защита от утечек, управление доступом.
Google Cloud Platform (GCP)	★★★★★	★★★★☆	★★★☆☆	★★★★★
	Полная поддержка авто-масштабирования и распределения нагрузки с высоким уровнем гибкости.	Глобальная сеть с гарантией 99.95% SLA, Multi-AZ, Multi-Region решения.	Скидки на долгосрочное использование, выгодные предложения для стартапов, плата за фактическое использование. Стоимость в среднем выше чем при использовании российских аналогов.	Соответствие международным стандартам GDPR, HIPAA, PCI DSS. Шифрование данных, защита от утечек, управление доступом.

Продолжение таблицы 1

VK Cloud	★★★★☆	★★★☆☆	★★★★★	★★★☆☆
	Фокусируется на локальных клиентах, что ограничивает глобальную масштабируемость.	Сеть в пределах РФ с гарантией 99.95% SLA, Multi-AZ решения.	Скидки на долгосрочное использование, выгодные предложения для стартапов, плата за фактическое использование. Гибкая система ценообразования, ориентированная на локальные условия.	Соответствие международному стандарту PCI DSS и №152-ФЗ. Шифрование данных, защита от утечек.
Yandex Cloud	★★★★☆	★★★★☆	★★★★★	★★★★☆
	Фокусируется на локальных клиентах, что ограничивает глобальную масштабируемость.	Сеть в пределах РФ с гарантией 99.99% SLA, Multi-AZ решения.	Скидки на долгосрочное использование, выгодные предложения для стартапов, плата за фактическое использование. Гибкая система ценообразования, ориентированная на локальные условия	Соответствие международным стандартам GDPR и PCI DSS. Соответствие №152-ФЗ. Шифрование данных, защита от утечек.

Выводы по первой главе

По завершению первого этапа исследования становится понятно, что облачные решения превосходят локальную инфраструктуру по ряду ключевых параметров, особенно в условиях быстроразвивающегося рынка интренет-продаж. Например, в периоды пиковых нагрузок (акции, распродажи, сезонные мероприятия) облако позволяет увеличивать или уменьшать ресурсы как вертикально, так и горизонтально, в течение нескольких минут, в то время как в локальной инфраструктуре на это могут уйти недели. Благодаря географически распределённым дата-центрам облачные провайдеры обеспечивают практически непрерывную работу. Это критично для интернет-магазинов, по специфике которых даже минуты простоя могут привести к значительным убыткам. Облачные решения позволяют избежать крупных капитальных затрат на покупку и обслуживание оборудования – вместо этого используется модель оплаты за фактическое использование.

Локальная инфраструктура больше подходит для компаний со специфическими требованиями безопасности и большими бюджетами. В то время как для большинства интернет-магазинов облачные решения являются более разумным выбором. Таким образом, переход на облачную платформу становится оптимальным выбором для развития интернет-магазинов, особенно в условиях высоких требований к масштабируемости и доступности.

После проведенного в главе сравнения, мы можем сделать вывод, что выбор облачной платформы для проектирования архитектуры веб-приложения зависит от множества факторов, включая конкретные требования проекта, стратегию развития, местоположение конечных пользователей и бюджет. При сравнении облачных провайдеров можно выделить несколько ключевых различий в их функциональности и возможностях.

Amazon Web Services (AWS) и Microsoft Azure предлагают наибольшую гибкость, функциональность, масштабируемость и возможности для обеспечения высокой доступности. Эти платформы идеально подходят для

международных компаний или тех, кто планирует значительный рост на международных рынках.

Однако, при выборе между AWS и Azure, стоит учитывать следующие особенности:

- AWS предлагает более гибкую и экономически эффективную модель, предоставляя множество вариантов скидок и акций, что делает его универсальным инструментом для различных типов организаций;
- Microsoft Azure, в свою очередь, ориентирован на снижение расходов при работе в экосистеме Microsoft, что может быть преимуществом для компаний, уже использующих продукты Microsoft, но не всегда подходит для всех типов бизнеса.

При всех прочих равных предпочтение стоит отдать AWS, так как его гибкость и экономическая эффективность делают его более универсальным инструментом на мировом рынке.

Yandex Cloud и VK Cloud могут быть более выгодными для российских пользователей и небольших предприятий, ориентированных на локальный рынок. Однако они значительно уступают по функционалу и гибкости при необходимости масштабирования на международном уровне. Поэтому, несмотря на их преимущества для локальных пользователей, в контексте международной экспансии они не являются столь конкурентоспособными, как AWS и Azure.

Глава 2 Проектирование архитектуры высокодоступных и масштабируемых веб-приложений с использованием возможностей AWS

Проанализировав и сравнив основных провайдеров облачных услуг и их предложения, мы выяснили, что в контексте мастшабируемости, высокой доступности, экономической эффективности и безопасности для веб-приложения, выходящего на международный рынок, наиболее оптимальным выбором будет применение Amazon Web Services. Однако для достижения максимального эффекта недостаточно просто выбрать облачного провайдера. Необходимо также разработать архитектуру, которая будет сбалансирована с точки зрения производительности, надежности и затрат. Таким образом, перед нами ставится задача в построении наиболее эффективной архитектуры для интернет-магазина с использованием облачных решений предоставляемых компанией Amazon.

Основные вызовы, стоящие перед современными веб-приложениями, можно условно разделить на несколько категорий:

- сложность архитектуры: большинство приложений представляют собой сложные системы с большим количеством взаимозависимых компонентов, таких как серверы приложений, базы данных, API, системы мониторинга и логирования. Каждый из этих компонентов должен быть оптимизирован для обеспечения стабильной работы и высокой производительности в рамках единой системы; [10]
- управление динамическими нагрузками: интернет-магазины сталкиваются с переменными нагрузками, особенно в периоды акций, распродаж или праздничных кампаний. Для обеспечения бесперебойной работы системы необходимо гибкое управление вычислительными ресурсами;
- требования к отказоустойчивости: доступность приложения 24/7 становится стандартом в электронной коммерции. Даже кратковременный простой может привести к значительным потерям дохода и снижению лояльности клиентов;

– безопасность и защита данных: работа с конфиденциальной пользовательской информацией требует строгого соблюдения стандартов безопасности и защиты данных от утечек или атак. [11]

Как мы уже выяснили, одним из ключевых решений для достижения этих целей является использование облачных технологий, предоставляющих широкий спектр инструментов для проектирования высокодоступных и масштабируемых систем. Такие технологии включают в себя виртуализацию, контейнеризацию, автоматизированное масштабирование и отказоустойчивость, что позволяет разработчикам сосредоточиться на улучшении пользовательского опыта, а не на управлении инфраструктурой.

2.1 Возможности и функционал облачных сервисов AWS для обеспечения отказоустойчивости и масштабируемости веб-приложения

Облачные сервисы AWS предоставляют богатый набор инструментов для обеспечения отказоустойчивости и масштабируемости веб-приложений, что особенно актуально для интернет-магазинов с высокой нагрузкой и пиковыми периодами трафика. В основе архитектуры лежат продуманные механизмы, позволяющие минимизировать риски простоев и обеспечивать стабильность работы даже при сбоях оборудования, отказах зон доступности или резких скачках нагрузки. [3]

Критически важную роль в обеспечении высокой доступности играет глобальная инфраструктура AWS, которая включает в себя 108 зон доступности (Availability Zones, AZs), которые представляют собой независимые дата-центры с автономным энергоснабжением и сетями, в 34 географических регионах. Эта инфраструктура постоянно расширяется и уже сейчас объявлены планы по созданию новых 18 зон доступности и 6 регионов AWS.

Наличие упомянутой выше инфрастурктуры позволяет использование нескольких зон доступности одновременно, что является ключевым для обеспечения отказоустойчивости. Например, сервисы вроде Amazon RDS или

Amazon Dynamodb поддерживают Multi-AZ репликацию, которая обеспечивает автоматическое переключение на резервные экземпляры базы данных в случае сбоя. При этом максимальное допустимое время простоя системы после сбоя или аварии (Recovery Time Objective, RTO) в случае такого переключения минимально – как правило, это занимает менее одной минуты. Резервирование базы данных в разных зонах доступности гарантирует, что при сбое одной из зон данные останутся доступными без ручного вмешательства. В интернет-магазинах это особенно важно для обеспечения доступности информации о заказах и пользователях. [30]

Неотъемлемым компонентом отказоустойчивости является Elastic Load Balancer (ELB), распределяющий входящие запросы между несколькими инстансами приложения. Например, Application Load Balancer (ALB) (одна из разновидностей ELB) идеально подходит для интернет-магазинов, так как поддерживает маршрутизацию запросов в зависимости от URL. [20] Это значит, что запросы на страницы каталога, корзины или процесса оформления заказа могут обрабатываться отдельными серверами, что повышает общую производительность. Для ALB Amazon гарантирует среднее время переключения между зонами доступности менее 1 секунды и пропускную способность в 100 000 соединений в секунду на один балансировщик. Наличие и правильная конфигурация такого сервиса как ELB позволит интернет-магазину обрабатывать десятки тысяч запросов в секунду, обеспечивая стабильную работу даже в условиях высокой нагрузки.

Для защиты от аппаратных сбоев AWS предлагает автоматическое восстановление (Auto Recovery). Например, при возникновении сбоя в работе экземпляра виртуального сервера (Elastic Compute Cloud, EC2) система автоматически запускает новый инстанс с сохранением конфигурации, это происходит благодаря работе сервиса по мониторинга CloudWatch. Это особенно полезно для ключевых сервисов интернет-магазина, например для таких, как обработка платежей.

Amazon ElastiCache предоставляет управляющий сервис для кэширования данных в облаке, что позволяет ускорить доступ к данным и улучшить производительность приложений. ElastiCache поддерживает два популярных движка: Redis и Memcached. Эти решения идеально подходят для кэширования данных, таких как результаты частых и идентичных запросов к базе данных, сессии пользователей или популярные товары в интернет-магазине, что позволяет значительно снизить нагрузку на систему, улучшая время отклика и обеспечивая более быструю обработку запросов. ElastiCache обеспечивает отказоустойчивость благодаря возможностям репликации и автоматического восстановления. [21] В случае сбоя одного узла ElastiCache переключится на резервный, минимизируя время простоя. Для Redis, например, можно настроить мастера с репликами в разных зонах доступности, что обеспечит отказоустойчивость на уровне кэширования данных. Также ElastiCache поддерживает масштабирование как по объему хранимых данных, так и по числу одновременных подключений, обеспечивая необходимую пропускную способность для обработки пиковых нагрузок.

Масштабируемость в AWS обеспечивается за счет горизонтального или вертикального масштабирования

- вертикальное масштабирование – увеличение вычислительных ресурсов одного сервера;
- горизонтальное масштабирование – добавление новых серверов для обработки нагрузки.

Auto Scaling Groups (ASG) отслеживают метрики, такие как использование CPU или количество активных подключений, и автоматически увеличивают или уменьшают количество или мощность серверов. Это критически важно для интернет-магазинов, которые могут столкнуться с резкими скачками трафика, например, во время распродаж или акций. Auto Scaling позволяет масштабировать инфраструктуру за считанные минуты, обрабатывая увеличение нагрузки в десятки раз. В пиковые периоды, такие как «Черная пятница», интернет-магазин может обслуживать тысячи пользователей одновременно, а в

периоды низкой пользовательской активности снизить ресурсы до минимума, экономя на затратах. [34]

Для более гибкого подхода к масштабированию активно используются серверлесс-технологии, это модель вычислений, которая позволяет разработчикам фокусироваться исключительно на написании кода, оставляя управление инфраструктурой облачному провайдеру. Термин "serverless" не означает отсутствие серверов; напротив, серверы используются, но их настройка, управление и масштабирование полностью абстрагированы от пользователя.

Основой serverless является модель "Function as a Service" (FaaS), где функции запускаются в ответ на события, такие как HTTP-запросы, изменения данных в базе, сообщения в очереди или триггеры на основе расписания. При этом модель оплаты строится по принципу pay-as-you-go, то есть пользователи платят только за фактическое время выполнения функций и объём обработанных данных. [19]

AWS Lambda является ключевым инструментом серверлесс-архитектуры в AWS. Она позволяет запускать функции на различных языках программирования, таких как Python, Node.js, Java и Go. Использование лямбда функций особенно востребовано в микросервисной архитектуре, где отдельные компоненты приложения могут масштабироваться независимо друг от друга. Сервис автоматически масштабируется для обработки большого числа параллельных запросов без необходимости управления серверами. Например, в интернет-магазине Lambda может использоваться для обработки следубщих событий: добавление товара в корзину, расчет скидок или отправка уведомлений клиентам. Такая архитектура позволяет поддерживать высокий уровень производительности при минимальных затратах на инфраструктуру.

Для хранения данных AWS предоставляет такие мощные решения как Amazon DynamoDB - serverless NoSQL БД формата ключ-значение, и Amazon Aurora - реляционная БД, совместимая с MySQL и PostgreSQL. DynamoDB поддерживает автоматическое масштабирование операций чтения и записи, что

особенно важно для интенсивно используемых сервисов, таких как корзина или каталог товаров. DynamoDB может обрабатывать до 10 трлн. запросов в сутки, а средняя задержка для 99% запросов не превышает 10 мс. Amazon Aurora, в свою очередь, автоматически масштабирует объем хранилища от 10 ГБ до 128 ТБ и поддерживает до 15 реплик с минимальной задержкой, что делает её оптимальным выбором для систем управления заказами. [33]

Глобальная доступность и ускорение контента достигаются с помощью Amazon CloudFront, глобальной сети доставки контента (CDN). CloudFront кэширует статический контент, такой как изображения товаров или файлы CSS/JS, в более чем 600 точках присутствия (Point of Presence, POP) по всему миру. Это сокращает время загрузки страниц для пользователей на 50–70%, что критически важно для улучшения пользовательского опыта. Например, пользователь из Азии, посещающий интернет-магазин с серверами в Европе, будет загружать изображения из ближайшего дата-центра, скажем, в Сингапуре.

Мониторинг и анализ производительности реализуются через Amazon CloudWatch и AWS X-Ray. CloudWatch отслеживает ключевые метрики, такие как задержка запросов или объем операций ввода-вывода, и позволяет настроить оповещения, например, при увеличении времени ответа API более чем на 500 мс. AWS X-Ray помогает анализировать трассировку запросов через всю архитектуру, от API до БД, выявляя узкие места, что особенно полезно для крупных интернет-магазинов с множеством интеграций, таких как платежные шлюзы или сторонние сервисы доставки. [15]

Построение архитерктуры с использованием этих инструментов обеспечит интернет-магазинам не только стабильность и производительность, но и гибкость, позволяя адаптироваться к изменениям нагрузки и минимизировать операционные издержки. Например, использование Auto Scaling и serverless-архитектуры позволяет сократить расходы на инфраструктуру на 30–40% в сравнении с традиционными подходами. Кроме того, такие решения как Amazon Cognito облегчают управление пользователями, обеспечивая безопасную регистрацию и аутентификацию.

Таким образом, правильные конфигурация и использование облачных сервисов AWS позволяет строить высокоэффективные интернет-магазины, способные справляться с самыми сложными вызовами современной электронной коммерции.

2.2 Обзор существующих подходов к созданию архитектур приложений в облаке с использованием AWS. Анализ их преимуществ и недостатков

Разнообразие задач и требований современной электронной коммерции диктует необходимость выбора оптимальных подходов к проектированию архитектуры веб-приложений. Как мы увидели в ходе исследования, AWS предлагает множество решений, которые можно адаптировать под различные сценарии использования. Основные подходы к созданию архитектур в AWS включают: монолитные архитектуры, микросервисные архитектуры и serverless-подход. [28]

В данном разделе будут подробно рассмотрены перечисленные подходы, а также проведён анализ их преимуществ и недостатков. Это позволит определить, какие из них будет наиболее эффективным для реализации высоконагруженных и масштабируемых интернет-магазинов с мировой аудиторией.

1. Монолитные архитектуры (Lift and Shift)

Перенос монолитных приложений в облако с помощью подхода Lift-and-Shift – это процесс миграции приложений без значительных изменений архитектуры. Этот метод обычно включает использование EC2 инстансов для хостинга и RDS для баз данных, что позволяет быстро перенести приложение в облако. Однако у этого подхода есть свои ограничения и нюансы.

Среди очевидных приемуществ стоит отметить быстрый выход на рынок - перенос монолита в AWS позволяет быстро максимально развернуть приложение без значительных изменений кода. [25] Это важно для интернет-

магазина, который стремится оперативно выйти на международный рынок, минимизируя время простоя и затраты на разработку новой архитектуры.

Также этот подход позволяет получить относительно высокую доступность с минимальными усилиями, благодаря встроенным механизмам для повышения доступности, таким как Auto Scaling, Elastic Load Balancing и RDS Multi-AZ Deployment, которые можно несложно интегрировать даже с монолитным приложением. [10]

Другое приемущество – масштабируемость под изменяющиеся нагрузки. Горизонтальное масштабирование возможно через добавление новых EC2-инстансов, хотя и требует предварительной настройки. При выходе на международный рынок магазин может масштабироваться по регионам, используя сервисы Amazon CloudFront для ускорения доставки контента. Для пиковых нагрузок, таких как распродажи или праздничные периоды, можно временно увеличить ресурсы, а затем сократить их, что повышает экономическую эффективность.

Краеугольным камнем данного похдода является тот факт, что подход экономически эффективен только на этапе миграции, но не на долгосрочной основе. Хотя Lift-and-Shift и не требует капитальных вложений в рефакторинг приложения и может использовать модель Pay-as-You-Go, оплачивая только используемые ресурсы, но поддержание большого числа EC2-инстансов в разных регионах может стать дорогостоящим, особенно по сравнению с микросервисной или серверлес-архитектурой. [23]

К недостаткам подхода также стоит отнести и ограниченную глобальную доступность - хотя AWS предоставляет инфраструктуру в разных регионах, монолитное приложение не может легко распределять нагрузку между ними, что может привести к задержкам для пользователей из удалённых регионов.

При росте продукта и числа пользователей следует ожидать появление трудностей с адаптацией. Монолиту будет сложнее поддерживать стабильность, поскольку такие архитектуры не предназначены для бесконечного горизонтального масштабирования.

2. Микросервисная архитектура

Микросервисная архитектура представляет собой подход к разработке приложений, при котором они разбиваются на небольшие, независимые компоненты, именуемые микросервисы. Эти микросервисы взаимодействуют между собой через четко определённые API. В контексте интернет-магазина, выходящего на международный рынок, использование микросервисной архитектуры имеет как преимущества, так и недостатки. Рассмотрим их подробно.

Каждый микросервис может масштабироваться независимо, это даёт нам высокую масштабируемость. Например, модуль корзины или обработки заказов можно масштабировать отдельно, если эти функции активно используются в пиковые периоды. AWS Fargate и EKS позволяют масштабировать контейнеры на основе реальной нагрузки, что особенно полезно для глобальных проектов.

Сервисы веб-приложения могут быть развернуты в нескольких регионах, что снижает вероятность полной остановки приложения при сбоях в одном из регионов. Использование AWS Aurora Global Database позволяет синхронизировать данные между регионами с задержкой менее 1 секунды, что критично для транзакционных операций интернет-магазина. Отказ одного микросервиса не приводит к остановке всего приложения, так как остальные продолжают работать независимо. Например, сбой в сервисе аналитики не повлияет на выполнение заказов. Таким образом данными обеспечивается высокая доступность и устойчивость к сбоям. [6]

Микросервисная архитектура дает гибкость разработки и развертывания, это достигается тем, что разные команды могут работать над своими микросервисами параллельно, что ускоряет разработку и внедрение новых функций.

Использование мульти-региональных развертываний, позволяет размещать сервисы ближе к пользователям, что улучшает производительность приложения и снижает задержки, обеспечивая поддержку глобальных пользователей.

Говоря о недостатках, микросервисная архитектура сложна для управления и отладки, так как требует сложной инфраструктуры, включая оркестрацию контейнеров, мониторинг, управление сетями и взаимодействие между сервисами. Для международного проекта это ещё сложнее из-за необходимости синхронизации данных между регионами. Логирование и мониторинг микросервисов требуют специальных инструментов, таких как AWS CloudWatch, Splunk или Datadog. Это усложняет диагностику проблем, особенно в распределённых системах.

Также всегда нужно помнить о том, что при использовании данного подхода увеличиваются расходы на сетевые взаимодействия между микросервисами, особенно если они находятся в разных регионах. [7] Например, при использовании Amazon Aurora Global Database в сочетании с AWS EKS расходы на передачу данных между регионами будут значительными.

Вопрос обеспечения безопосности является очень важным для любой организации работающей в сфере электронной коммерции. Сложность в этом дает то, что каждый микросервис требует отдельной конфигурации безопасности. Для интернет-магазина, работающего с личными данными и платежной информацией, это будет непростым вызовом.

3. Serverless-архитектура

Данный вид архитектуры использует такие сервисы AWS как Lambda, API Gateway, DynamoDB, CloudFront и другие, для создания высокомасштабируемых приложений с минимальными затратами на управление инфраструктурой. В серверлес-архитектуре, ресурсы автоматически масштабируются в зависимости от объёма запросов, что подходит для интернет-магазинов, которые выходят на международный рынок.

Serverless-архитектура обеспечивает высокую производительность и масштабируемость. AWS Lambda, являясь ядром подхода «бессерверных» вычислений, автоматически масштабирует приложение в ответ на количество запросов, что позволяет эффективно справляться с пиками трафика, например, в периоды акций или праздников, без необходимости заранее развертывать

дополнительные серверы. Система может обслуживать более чем 100 тыс. одновременных пользователей, так как Lambda автоматически увеличивает количество экземпляров функции, чтобы соответствовать нагрузке. В сочетании с AWS DynamoDB, которая поддерживает масштабирование до более чем 1 млн. запросов в секунду с минимальными затратами на настройку и управление базой данных, эта архитектура становится особенно эффективной для интернет-магазинов с растущей международной клиентской базой, обеспечивая высокую производительность и гибкость. [27]

Экономическая эффективность серверлес-архитектуры заключается в том, что вы платите только за время выполнения Lambda-функций, что делает этот подход особенно привлекательным для интернет-магазинов. В отличие от традиционных выделенных серверов, где приходится оплачивать постоянные ресурсы, которые могут не использоваться в обычные дни, серверлес-архитектура позволяет избежать таких затрат. Например, стоимость исполнения одной функции Lambda составляет примерно $0.0000002 за каждую миллисекунду выполнения, что значительно дешевле по сравнению с традиционными выделенными серверами. Дополнительно AWS предоставляет бесплатный уровень, включающий 1 млн. бесплатных запросов и 400,000 GB-секунд вычислений в месяц, что позволяет существенно экономить на начальном этапе развития бизнеса.

Благодаря инфраструктуре AWS, которая гарантирует отказоустойчивость и распределённость в различных регионах обеспечивается высокая доступность лямбда-функций. Функции могут выполняться одновременно в нескольких регионах, что значительно снижает вероятность потери доступности для пользователей в разных частях мира. Дополнительно, CloudFront снижает задержки, предоставляя быстрый доступ к контенту для пользователей в разных странах, что не только улучшает пользовательский опыт, но и увеличивает скорость обработки запросов. Это в свою очередь способствует обеспечению стабильной работы приложения, независимо от географического расположения пользователей.

Безопасность в AWS Lambda обеспечивается интеграцией с IAM (Identity and Access Management), что позволяет задавать чёткие роли и права доступа для каждой функции, обеспечивая безопасную работу с данными. Кроме того, AWS предлагает шифрование данных как на уровне хранения (например, с использованием KMS для DynamoDB), так и на уровне передачи данных, что гарантирует защиту информации на всех этапах её обработки и хранения. Это повышает уровень безопасности, предотвращая несанкционированный доступ и утечку данных. [32]

Конечно же, этот подход тоже не лишён недостатков. У сервиса AWS Lambda есть ряд технических ограничений.

Например, лямбда функции имеют ограничение в 15 минут на выполенение одной задачи. Если функция не успевает завершить выполнение в этот промежуток, она будет автоматически прервана, и процесс будет завершён с ошибкой. Это может стать проблемой для интернет-магазинов, которые выполняют длительные операции, такие как многоступенчатая сложная обработка заказов или интеграция с внешними сервисами, где требуется больше времени для выполнения задачи. [18] Однако для большинства операций в электронной коммерции, таких как обработка транзакций, создание заказов, их валидация и другие стандартные процессы, этого времени достаточно.

AWS Lambda не сохраняет состояние между вызовами, что означает, что для выполнения сложной логики, связанной с состоянием заказа или корзины, необходимо использовать внешние сервисы, такие как DynamoDB или S3. В условиях высокой нагрузки или при сложных рабочих процессах, например, при сложной валидации товаров или интеграции с внешними платёжными системами, отсутствие поддержания состояния будет очевидной проблемой. Для эффективного управления состоянием в таких ситуациях требуется дополнительная координация между сервисами, что увеличивает сложность архитектуры. Важно также учитывать, что при высоких нагрузках, когда количество вызовов Lambda-функций увеличивается, правильная настройка и

управление состоянием могут потребовать дополнительных усилий для обеспечения надёжности и эффективности работы системы.

Есть существенный риск того, что длительные запросы, касающиеся взаимодействия с внешними системами, могут не успеть обработаться. Например, если запрос из браузера пользователя требует продолжительной обработки (сложная транзакция в интернет-магазине), или если клиент должен ожидать длительный отклик от системы, это может привести к тайм-аутам на уровне API Gateway, несмотря на то, что сама Lambda может работать дальше. Это существенно повысит стоимость использования сервиса, поскольку затраты на Lambda зависят от нескольких факторов, включая время работы функции. В API Gateway существует ограничение на время ожидания, которое составляет 29 секунд для обработки внешнего запроса, после этого срок ожидания истекает, и клиент получает ошибку. В процессе покупки могут происходить многократные обращения к базе данных и внешним сервисам, это может требовать длительного времени на выполнение или последовательности шагов, которые лучше реализовать в более традиционных подходах с постоянными серверами, например - EC2. [29]

Для успешной реализации веб-приложения, способного обеспечить высокую доступность и масштабируемость на международном уровне, крайне важно учитывать определить оптимальный подход к организации инфраструктуры. Разнообразие подходов, таких как монолитные, микросервисные и серверлес-архитектуры, позволяют учитывать различные нужды бизнеса, однако каждый из этих подходов имеет свои ограничения.

Как показал анализ, традиционные решения, такие как Lift-and-Shift или микросервисные архитектуры, хотя и могут обеспечить определенную гибкость, сталкиваются с проблемами, связанными с экономической эффективностью, сложностью масштабирования и глобальной доступностью. Для международных проектов, где важны высокая доступность и гибкость, а также оптимизация затрат на длительную перспективу, требуется более комплексная и адаптивная инфраструктура.

Наиболее эффективным решением для обеспечения высоконагруженных и масштабируемых интернет-магазинов будет создание гибридной архитектуры в AWS, которая сочетает в себе преимущества различных подходов. Такой подход не только обеспечит отказоустойчивость, но и позволит эффективно соответствовать растущим требованиям пользователей по всему миру, минимизируя затраты на инфраструктуру.

В дальнейшем мы рассмотрим более детально, как такая гибридная архитектура может быть реализована с использованием передовых решений AWS для достижения оптимального сочетания производительности, гибкости и экономической эффективности.

2.3 Разработка нового подхода к интеграции AWS в архитектуру веб-приложения для обеспечения высокой доступности и масштабируемости

Для решения обозначенных проблем в рамках исследования была разработана гибридная архитектура, сочетающая serverless-решения и управляемые сервисы AWS. Этот подход минимизирует ограничения каждой из технологий, оптимизируя доступность и масштабируемость, при этом сохраняя экономическую эффективность и безопасность. На рисунке 1 представлены основные компоненты новой архитектуры с использованием сервисов AWS и их взаимосвязь.

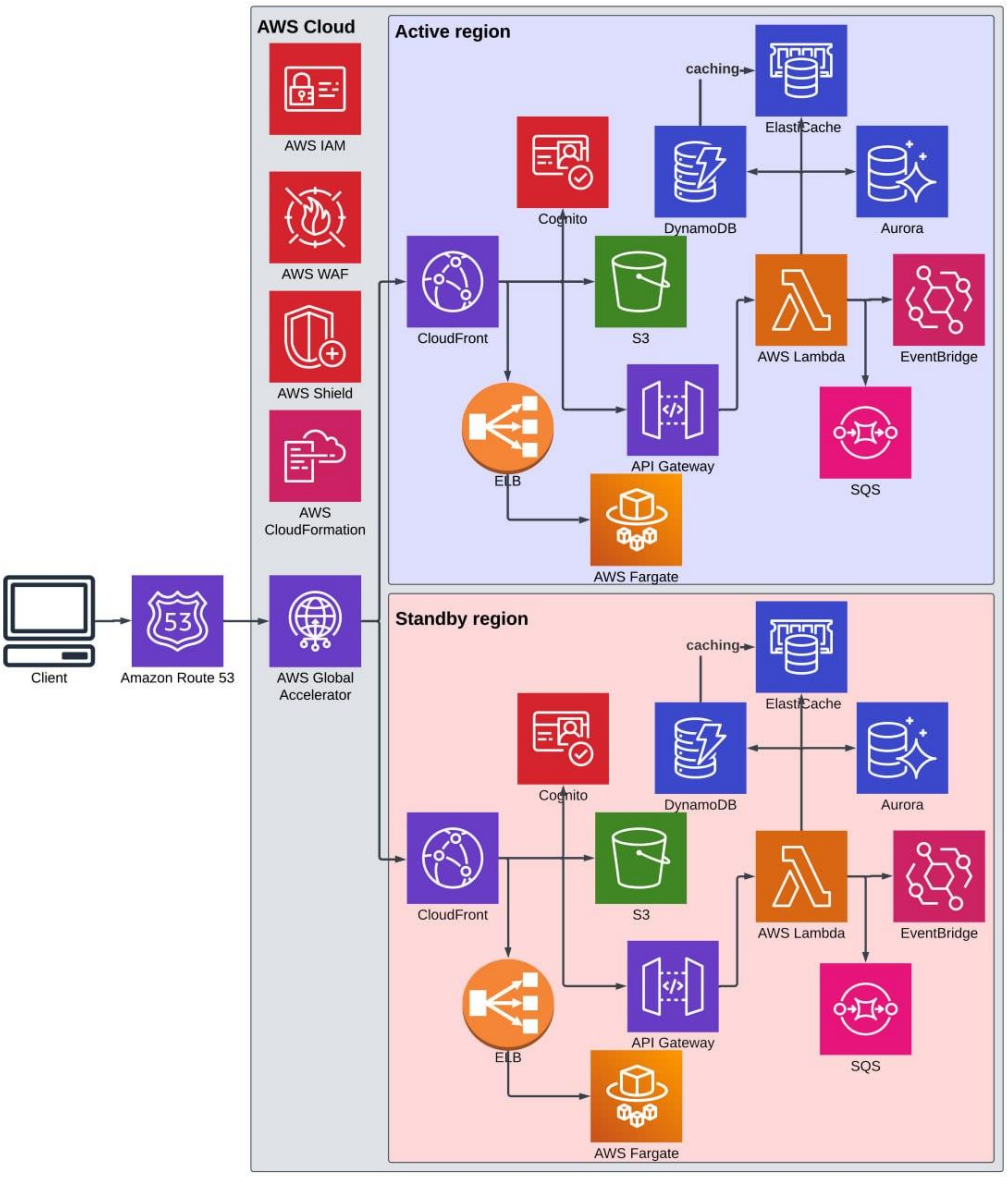

Рисунок 1 – Схема разработанной гибридной архитектуры интернет-магазина с применением облачных сервисов AWS для обеспечения высокой доступности и масштабируемости

Причины использования serverless-подхода как основы архитектуры для интернет-магазина обусловлены следующими факторами:

– экономическая эффективность – платформа автоматически освобождает ресурсы после выполнения функции, устраняя необходимость содержать простаивающую инфраструктуру;

– гибкость разработки – Serverless подходит для создания модульных приложений, что упрощает внедрение изменений и улучшений;

– скорость развертывания – благодаря минимальной зависимости от инфраструктуры разработчики могут быстрее тестировать и выпускать обновления.

Ключевые особенности новой архитектуры – это глобальная доступность через Multi-Region Deployment, ведь использование AWS Global Accelerator для интеллектуальной маршрутизации трафика между регионами сможет минимизировать задержки. Также, Active-Standby Region Deployment снижает затраты и поддерживает высокую отказоустойчивость, ведь основной регион обрабатывает весь пользовательский трафик, а резервный регион настроен для быстрого переключения в случае сбоя через Route 53 Health Checks и Global Accelerator.

Гибридное масштабирование реализовано с помощью AWS Lambda, которая используется для API-запросов и микросервисов, Amazon API Gateway, который управляет вызовами API и маршрутизирует их на соответствующие Lambda-функции, AWS Fargate, обеспечивающим выполнение контейнеров для более долгосрочных и ресурсоемких задач, таких как массовая обработка заказов и Amazon SQS и Amazon EventBridge, которые используются для обработки событий и очередей сообщений между компонентами приложения.

Масштабируемая база данных с автоматической репликацией основана на использовании Amazon Aurora Global Database – благодаря этому главная база данных синхронизируется между регионами менее чем за секунду. Amazon DynamoDB также используется для хранения горячих данных, таких как каталог товаров, с активным кэшированием через Amazon ElastiCache (Redis) для минимизации задержек, а Amazon S3 выступает в роли хранилища для архивных данных и статических файлов.

Автоматическое управление инфраструктурой обеспечивается работой AWS CloudFormation для создания и управления всей инфраструктурой как кодом. Elastic Load Balancer обеспечивает распределение трафика между

сервисами, а Auto Scaling Groups управляют горизонтальным масштабированием EC2-инстансов в случае увеличения нагрузки.

Экономическая эффективность достигается за счет комбинации таких сервисов, как spot-инстансы, используемые для выполнения аналитических или не критичных задач, и Savings Plans наряду с Reserved Instances для оптимизации затрат для Aurora.

Интеграция с CDN и безопасностью реализована с помощью Amazon CloudFront для доставки статического контента (CSS, JS) для пользователей с минимальной задержкой, а также AWS WAF и Shield Advanced, которые обеспечивают надежную защиту от DDoS-атак и управление правилами безопасности на уровне приложений. AWS IAM гарантирует соблюдение политики минимизации прав доступа для сервисов и пользователей, а AWS Cognito реализует аутентификацию пользователей для клиентских приложений.

Выводы по второй главе

Использование разработанной в рамках исследования архитектуры является наиболее эффективным для интернет-магазина, осуществляющего экспансию на международный рынок, для достижения высокой доступности и масштабируемости с оптимальными экономической эффективностью и безопасностью. Это связано со следующими факторами:

- высокая отказоустойчивость – если один регион становится недоступным, система автоматически переключается на другой, поддерживая 99.99% доступности;
- масштабируемость на международный рынок гарантирована благодаря поддержке более 1 млн запросов в секунду через API Gateway и Lambda, что идеально подходит для крупных распродаж, таких как «Черная пятница», а быстрая доставка контента пользователям из любой страны будет реализована через сервис CloudFront;
- экономия затрат реализована за счет оптимизации расходов на инфраструктуру за счет serverless и резервных ресурсов и минимизации расходов на хранение через кэширование и использование S3 для архивных данных;
- повышенная безопасность обеспечена интеграцией с AWS Cognito для безопасной аутентификации пользователей и минимизацией прав доступа через IAM-политики.

Предложенная гибридная архитектура AWS оптимально сочетает отказоустойчивость, масштабируемость, экономическую эффективность и безопасность, что особенно важно для интернет-магазинов, стремящихся выйти на международный рынок. Разработанная архитектура легко адаптируется под технические и экономические ограничения, становясь универсальным выбором для современных веб-приложений направления «электронная коммерция».

Глава 3 Оптимизация бизнес-процесса продажи оборудования с использованием сервисов AWS для работы веб-приложения

В первую очередь был проведен анализ деятельности компании ООО «Тотал Продакшн» и выделены основные рабочие процессы. В ходе данного этапа были использованы подходы бизнес-анализа, такие как изучение основных направлений деятельности, их взаимодействие, а также выявление ключевых процессов и ресурсов, используемых в работе компании.

ООО «Тотал Продакшн» предоставляет широкий спектр услуг в области сценического и продакшн производства. Компания специализируется на продаже концертного и сценического оборудования, организации мероприятий различного масштаба, включая концерты, корпоративы, фестивали и другие события «под ключ». Имея современную техническую базу и команду профессионалов, компания активно работает на локальном рынке и планирует выход в продажи на международном уровне.

Основные рабочие процессы ООО «Тотал Продакшн» сосредоточены вокруг продажи, аренды, транспортировки, монтажа оборудования и управления проектами мероприятий. Существенную роль играют процессы логистики и оплаты и обслуживания оборудования, а также информационная поддержка, которая становится особенно важной в связи с планами компании по запуску интернет-магазина на международный рынок.

В целях апробации гипотезы исследования и оптимизации инфраструктуры веб-приложения интернет-магазина ООО «Тотал Продакшн» в инфраструктуру приложения были интегрированы облачные технологии AWS для обеспечения высокой доступности, гибкости масштабирования и минимизации рисков, связанных с техническими сбоями и простоем системы.

Проект направлен на оптимизацию IT-инфраструктуры компании ООО «Тотал Продакшн» с целью обеспечения её масштабируемости и высокой доступности в условиях увеличения пользовательской нагрузки и запуска интернет-магазина на международный рынок.

Основные бизнес-цели проекта:

1. Обеспечение высокой доступности веб-приложения

Обеспечить доступность веб-приложения на уровне 99.99% в условиях пиковых нагрузок, минимизируя время его простоя

2. Предоставление возможности масштабирования IT-инфраструктуры

Предоставить возможность автоматического увеличения ресурсов при росте пользовательской активности и уменьшения при снижении в целях поддержки гибкости системы для работы с различными объемами данных.

3. Снижение затрат на содержание IT-инфраструктуры

Оптимизировать расходы на IT-инфраструктуру путем перехода от локальных серверов к облачной модели, с оплатой только за фактическое использование ресурсов.

4. Выход на международный рынок

Обеспечить техническую базу для масштабирования бизнеса, выходя с интернет-магазином на международный рынок; удовлетворение требований клиентов в разных регионах за счет быстрого и надежного доступа к системе.

5. Повышение безопасности хранения данных

Гарантировать защиту пользовательских данных и соответствие международным стандартам безопасности.

Объем проекта с точки зрения технического процесса:

1. Анализ текущей инфраструктуры и её недостатков;
2. Создание инфраструктуры облачных сервисов AWS, которые обеспечат надежность и масштабируемость системы по архитектуре разработанной в процессе исследования;
3. Миграция веб-приложения с локального сервера в облако AWS;
4. Тестирование системы для проверки производительности и доступности;
5. Настройка механизмов резервного копирования и восстановления данных;
6. Документация и обучение сотрудников для поддержания работы облачной инфраструктуры.

На рисунке 2 обозначено дерево целей проекта, которое наглядно демонстрирует, как главная цель проекта связана с различными аспектами оптимизации IT-инфраструктуры компании, обеспечивая решение всех ключевых бизнес-задач для роста и развития ООО «Тотал Продакшн».

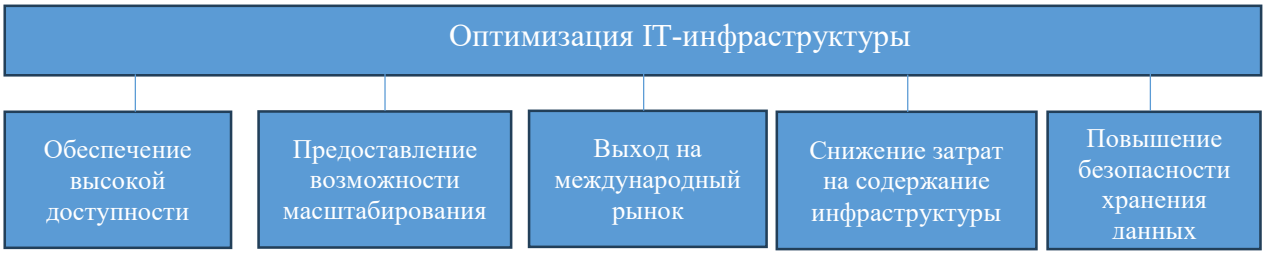

Рисунок 2 – Схема дерева целей проекта

3.1 Определение ключевого бизнес-процесса для проведения оптимизации

При анализе бизнес-процессов в контексте функционирования интернет-магазина заказчика ООО "Тотал Продакшн" и его инфраструктуры в качестве ключевого бизнес-процесса был определен процесс создания и обработки заявок клиентов (рисунок 3):

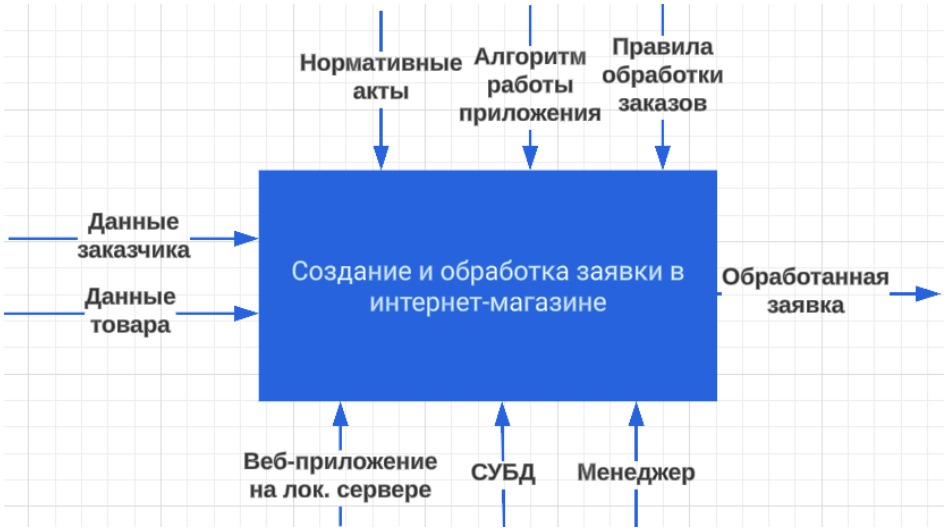

Рисунок 3 – Контекстная диаграмма «КАК ЕСТЬ» бизнес-процесса «Создание и обработка заявки в интернет-магазине», разработанная в методологии IDEF0

На контекстном уровне процесс «Создание и обработка заявки в интернет-магазине» выглядит следующим образом:

- вход: данные заказчика, данные товара;
- выход: обработанная заявка;
- механизмы: веб-приложение на локальном сервере, СУБД, менеджер организации;
- контроль: нормативные акты, алгоритм работы приложения, правила обработки заказов.

Этот процесс может быть декомпозирован на следующие подпроцессы:

1. Клиент делает заказ через веб-приложение, которое принимает запрос и проводит первичную валидацию данных заказа.
   а) процесс: принятие и валидация запроса о создании новой заявки;
   б) вход: данные заказчика, данные товара;
   в) выход: проверенная и валидная заявка;
   г) механизмы: веб-приложение на локальном сервере;
   д) контроль: алгоритм работы приложения, нормативные акты.
2. После валидации, заявка добавляется в базу данных с помощью встроенных функций веб-приложения.
   а) процесс: добавление новой заявки в БД;
   б) вход: проверенная и валидная заявка;
   в) выход: заявка, добавленная в БД;
   г) механизмы: веб-приложение на локальном сервере, СУБД;
   д) контроль: алгоритм работы приложения, нормативные акты.
3. После добавления заявки в базу данных, система автоматически отправляет уведомление менеджеру о новой заявке.
   а) процесс: уведомление менеджера о новой заявке;
   б) вход: заявка, добавленная в БД;
   в) выход: уведомление для менеджера;
   г) механизмы: веб-приложение на локальном сервере;

д) контроль: алгоритм работы приложения, нормативные акты.

4. Менеджер, получив уведомление, приступает к обработке заказа в соответствии с внутренними правилами и нормативами.
   а) процесс: обработка новой заявки менеджером;
   б) вход: уведомление для менеджера, заявка из БД;
   в) выход: обработанная заявка;
   г) механизмы: менеджер;
   д) контроль: правила обработки заказов, нормативные акты.

На рисунке 4 представлена декомпозиция указанного бизнес-процесса в плоскости «КАК ЕСТЬ», в котором комплексный процесс создания и обработки заявки в интернет-магазине разбивается на более мелкие взаимосвязанные между собой процессы:

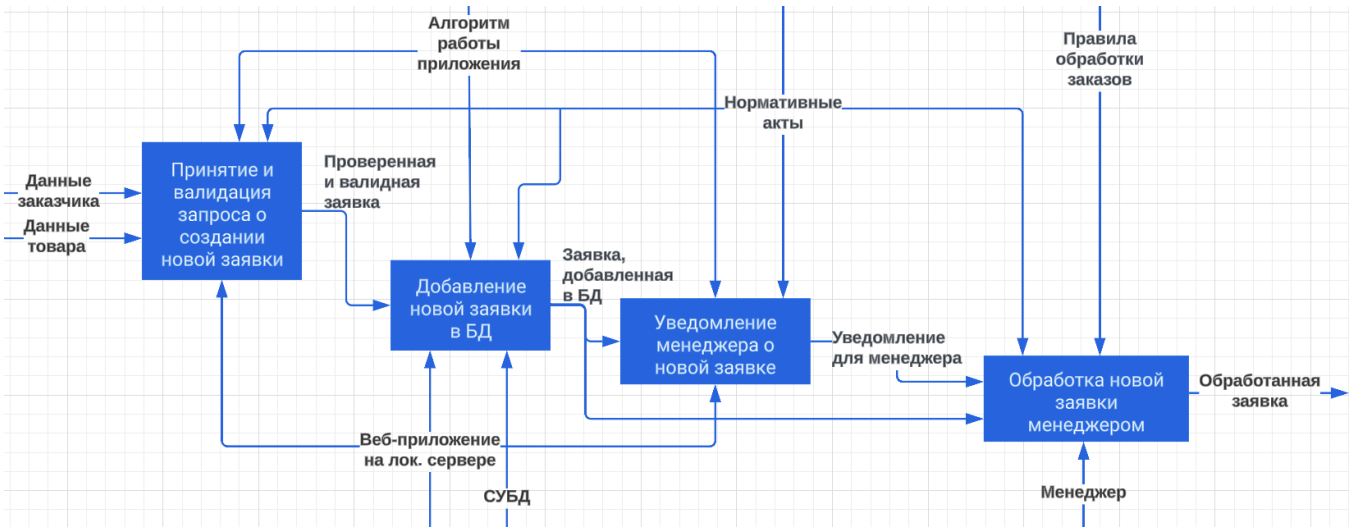

Рисунок 4 – Декомпозиция контекстной диаграммы «КАК ЕСТЬ» бизнес-процесса «Создание и обработка заявки в интернет-магазине», разработанная в методологии IDEF0

На основании анализа бизнес-процесса «Создание и обработка заявки в интернет-магазине» можно выделить следующие точки для его улучшения:

– масштабируемость веб-приложения: с учетом стратегической цели компании по выводу интернет-магазина на международный рынок, важно обеспечить возможность быстрого масштабирования веб-приложения в ответ на увеличение пользовательской нагрузки;

– высокая доступность веб-приложения: важно обеспечить непрерывное функционирование веб-приложения, минимизируя время простоя и перерывы в работе, что можно достичь путем внедрения технологий отказоустойчивости и резервного копирования данных;

– оптимизация процесса обработки заявок: существует потенциал для автоматизации некоторых этапов обработки заявок, что может сократить время обработки и уменьшить вероятность ошибок, например, можно рассмотреть возможность автоматизации проверки наличия товара на складе;

– обеспечение безопасности данных: с учетом увеличения объема обрабатываемых данных и расширения на международный рынок, важно обеспечить надежную защиту данных клиентов и соблюдение требований законодательства о защите персональных данных.

Эти улучшения соответствуют стратегическим целям компании, которые включают расширение интернет-магазина на международный рынок и увеличение его конкурентоспособности. Переход на облачную инфраструктуру и оптимизация процессов обработки запросов и масштабирования позволят улучшить качество обслуживания, увеличить доступность и производительность веб-приложения, что, в свою очередь, способствует достижению стратегических целей компании.

Текущий процесс создания и обработки заявки от клиента включает в себя следующие шаги:

1) Клиент выбирает интересующие товары и отправляет через веб-приложение запрос на создание заявки.
2) Веб-приложение, размещенное на локальном сервере, получает запрос. При наличии достаточных ресурсов запрос начинает обрабатываться, иначе сервер отвечает ошибкой.
3) Веб-приложение валидирует полученный запрос. В случае ошибок валидации, клиенту возвращается соответствующее сообщение.

4) Если запрос валиден, то он отправляется в базу данных, размещенную также локально. В базе данных, в случае её доступности, происходит сохранение заявки, в противном случае, сервер отвечает ошибкой и процесс создания заявки обрывается.
5) После сохранения заявки в базе данных, веб-приложение отправляет уведомление менеджеру.
6) Менеджер получает уведомление и начинает обработку заказа.

После тщательного анализа бизнес-процесса было установлено, что существующая инфраструктура веб-приложения, базирующаяся на локальном сервере в офисе компании, имеет ограниченный потенциал для расширения бизнеса и выхода на международный рынок. Это связано с следующими проблемами:

- если сервер не имеет достаточных ресурсов для обработки запросов пользователей, он перестаёт принимать входящие запросы, что приводит к негативному пользовательскому опыту, ухудшению репутации компании и потере потенциальной прибыли;
- веб-приложение или база данных могут стать недоступными по множеству причин, включая аппаратные и программные сбои, проблемы с производительностью, проблемы с сетью, проблемы с безопасностью, плановое обслуживание и человеческие ошибки.

На рисунке 5 существующий процесс отражен в виде диаграммы дорожек «КАК ЕСТЬ», что наглядно демонстрирует недостатки текущей инфраструктуры веб-приложения:

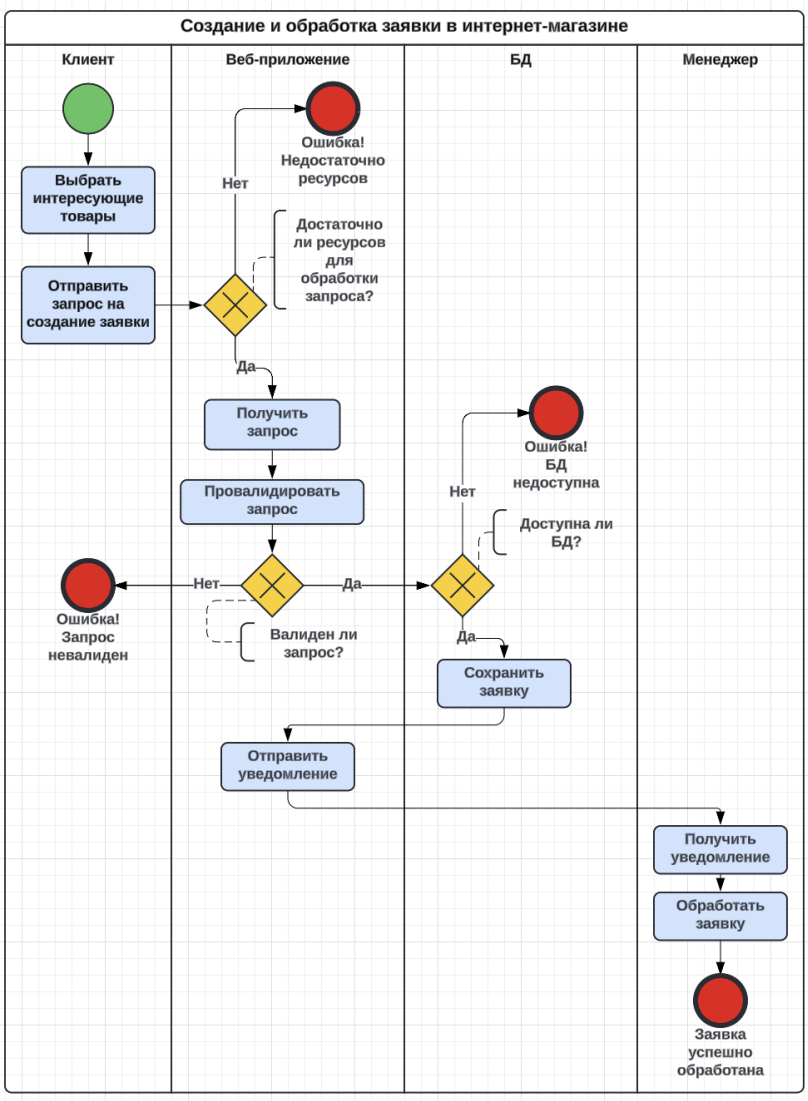

Рисунок 5 – Диаграмма дорожек «КАК ЕСТЬ» бизнес-процесса «Создание и обработка заявки в интернет-магазине», разработанная по методологии BPMN

Улучшенный процесс «Создание и обработка заявки в интернет-магазине» будет включать в себя следующие шаги:

1) Клиент выбирает интересующие товары и отправляет через веб-приложение запрос на создание заявки;

2) Веб-приложение, размещенное на облачной инфраструктуре, получает запрос, и, благодаря автоматическому масштабированию, приложение всегда имеет достаточно ресурсов для обработки запроса, даже при внезапном увеличении нагрузки;

3) Веб-приложение валидирует запрос, и, в случае ошибок валидации, клиенту возвращается соответствующее сообщение;

4) Если запрос валиден, то он отправляется в базу данных, размещенную в облаке, и благодаря высокой доступности облачной инфраструктуры, база данных всегда доступна для сохранения заявки (в случае возникновения проблем с основной базой данных, автоматически активируется резервная база данных);

5) После сохранения заявки в базе данных, веб-приложение автоматически отправляет уведомление менеджеру;

6) Менеджер получает уведомление и начинает обработку заказа.

Модель «БЫТЬ» демонстрирует, как после выявления потребностей бизнеса изменится бизнес-процесс «Создание и обработка заявки в интернет-магазине» в ООО «Тотал Продакшн» (рисунок 6):

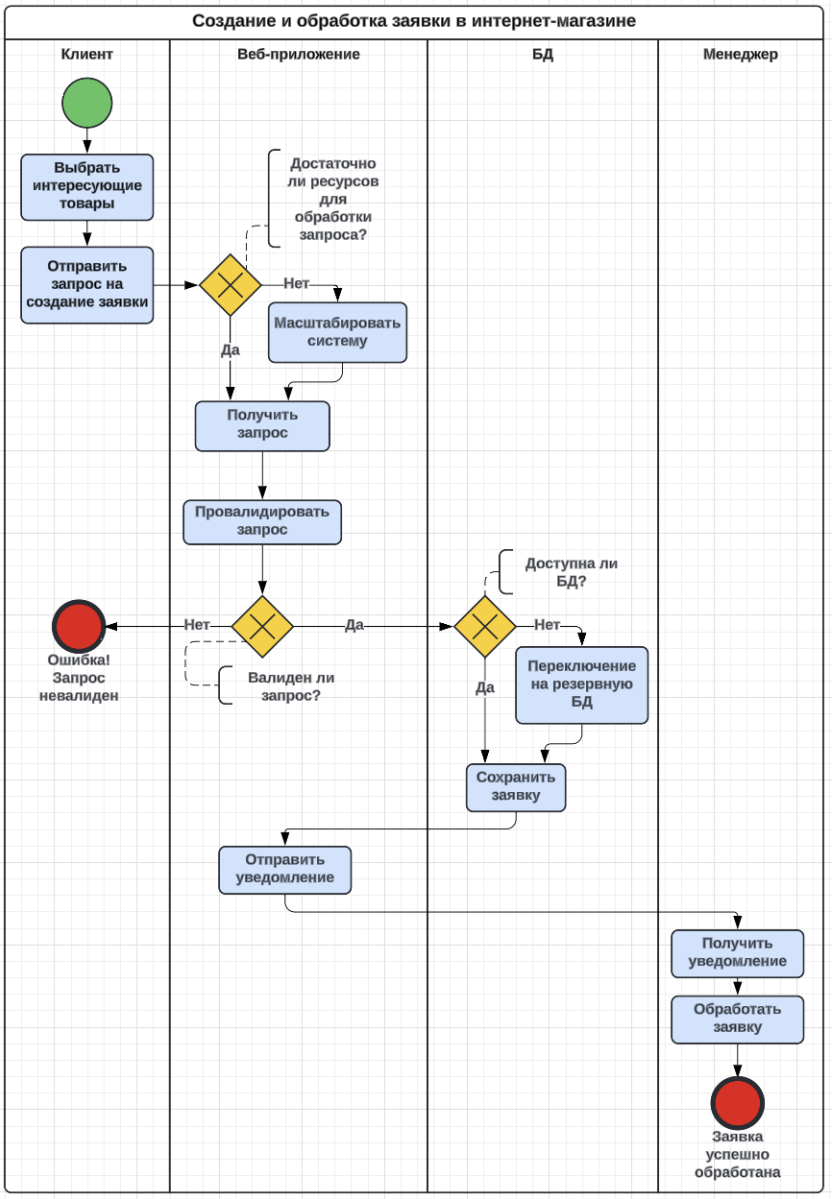

Рисунок 6 – Диаграмма дорожек «БЫТЬ» бизнес-процесса «Создание и обработка заявки в интернет-магазине», разработанная по методологии BPMN

Теперь этот процесс учитывает стратегические цели компании по расширению на международный рынок и увеличению конкурентоспособности. Благодаря использованию облачной инфраструктуры, интернет-магазин получает высокую доступность и масштабируемость, что позволяет обслуживать больше клиентов и быстро реагировать на изменения рынка. Автоматизация процессов обработки заявок помогает ускорить и упростить работу менеджеров, уменьшить вероятность ошибок и улучшить общее качество обслуживания.

3.2 Разработка экспериментального плана и апробация гипотезы исследования

Для апробации гипотезы исследования был разработан экспериментальный план, целью которого является подтверждение того, что если для проектирования системы интернет-магазина будут применены облачные технологии, то система сможет сможет эффективно масштабироваться в ответ на изменяющийся интернет-трафик, обеспечивая высокую доступность и отказоустойчивость, минимизируя простои и перерывы в обслуживании, а также сохранять оптимальную производительность при изменении нагрузки.

Для проведения эксперимента были выбраны следующие критерии, отображенные в таблице 2:

Таблица 2 – Критерии оценки успешности экспериментального внедрения облачной инфраструктуры в веб-приложение ООО «Тотал Продакшн»

Название	Критерий	Метод измерения	Цель
Доступность системы (%)	Уровень доступности системы для пользователей	Мониторинг показателя uptime системы и анализ времени отклика системы	Сравнить время безотказной работы до и после миграции
Время отклика (на 500 и на 1000 пользователей)	Производительность системы под разными уровнями нагрузки	Проведение нагрузочного тестирования	Измерить время отклика и максимальную нагрузку до и после миграции
Время реакции на изменение нагрузки (минуты)	Масштабирование ресурсов в ответ на увеличение нагрузки	Наблюдение за временем масштабирования и скоростью реакции на изменения нагрузки	Проверить способность системы автоматически масштабировать ресурсы

Продолжение таблицы 2

Частота и длительность простоев (случаи, минуты)	Наличие простоев и их продолжительность	Сбор и анализ логов и проверка отчетов о простоях	Оценить временя реагирования на сбои и их частоту

Для проверки гипотезы исследования были использованы указанные критерии, которые помогут оценить влияние перехода на AWS и автоматизацию бизнес-процессов на ключевые аспекты работы компании. Регулярный мониторинг и анализ данных по каждому из критериев позволит сделать выводы о корректности гипотезы и эффективности проведенных изменений.

Также был утвержден план проведения эксперимента:

Этап 1: Подготовка и тестирование исходной системы:

а) Собрать метрики для анализа текущего состояния веб-приложения;

б) Провести нагрузочное тестирование для выявления слабых мест.

Этап 2: Миграция инфраструктуры в AWS:

а) Перенести инфраструктуру в AWS, настроить конфигурацию доступности и автоматическое масштабирование.

Этап 3: Тестирование системы после миграции в AWS:

а) Повторить измерения метрик после миграции для сравнения результатов;

б) Оценить выполнение критериев.

Этап 4: Сравнительный анализ и апробация гипотезы:

а) Сравнить полученные результаты по каждому критерию;

б) Провести апробацию гипотезы

Если результаты покажут улучшение по всем критериям, гипотеза исследования будет считаться подтвержденной.

Этап 1: Подготовка и тестирование исходной системы

Для мониторинга уровня доступности системы использовалась платформа Zabbix, которая была развернута на локальном сервере ООО «Тотал Продакшн».

После установки была настроена Zabbix, начиная с выбора подходящих хостов и агентов для мониторинга ключевых показателей системы. На протяжении недели Zabbix постоянно собирал данные об uptime системы и задержках в ответе. В ходе эксперимента показатели проверялись вручную через веб-интерфейс Zabbix, а также через регулярное автоматическое создание отчетов по ключевым метрикам, чтобы каждый день иметь доступ к актуальной информации.

Сначала был проведен анализ существующей инфраструктуры, чтобы определить, какие компоненты критичны для работы веб-приложения. На основе этого анализа в Zabbix были добавлены узлы сети (сервер базы данных, веб-сервер и сетевые устройства) и настроены агенты Zabbix на каждом из них.

Настройка мониторинга доступности:

1. Определение узлов мониторинга: были выбраны узлы, которые являлись ключевыми точками в системе, а именно:
   - веб-сервер приложения;
   - база данных (сервер MySQL);
   - сетевые устройства, обеспечивающие связь между серверами.
2. Установка агентов Zabbix: на каждом узле был установлен агент Zabbix, который собирал данные о состоянии узла и передавал их на центральный сервер мониторинга. Агенты были настроены на сбор метрик, таких как загрузка процессора, использование памяти, сетевые задержки и время отклика.
3. Настройка триггеров и шаблонов: чтобы получать уведомления о проблемах в реальном времени, были настроены Zabbix триггеры. Если время отклика веб-сервера превышало 400 мс, система отправляла оповещение. Также были настроены триггеры для отслеживания простоев, использования ресурсов и ошибок на сервере базы данных. Были созданы шаблоны, которые автоматически применялись к добавленным узлам, что позволило унифицировать мониторинг.

– мониторинг uptime: система собирала информацию о доступности серверов в режиме реального времени, фиксируя моменты, когда сервера переставали отвечать или появлялись ошибки сети;
– измерение задержек: для этого были использованы встроенные средства Zabbix, которые пинговали серверы с установленным интервалом и записывали время отклика (оповещения приходили в случае, если задержка превышала заданный порог).

В конце недели были собраны и экспортированы все необходимые данные для анализа. Было важно проверить, чтобы данные об uptime и времени отклика были полными и корректными, поэтому нами несколько раз вручную были проверены логи в Zabbix, чтобы убедиться, что все узлы корректно передают данные. После этого данные были обработаны, по результатам был создал отчет, в котором мы указали средние значения доступности системы и время отклика.

2. Нагрузочное тестирование: для проведения нагрузочного тестирования использовалось программное обеспечение Apache JMeter, который является мощным инструментом для имитации реальной нагрузки на систему. Процесс начинался с подготовки сценариев тестирования и настройки JMeter для эмуляции пользователей, подключающихся к веб-приложению.

1. Подготовка сценария тестирования: был разработан сценарий, отражающий типичное поведение пользователей, включая такие действия, как переходы между страницами, выполнение поисковых запросов и отправка форм, что позволило создать нагрузку, близкую к реальной эксплуатации приложения (основной задачей на данном этапе было смоделировать взаимодействие с веб-приложением, охватывающее ключевые функции, такие как авторизация, просмотр продуктов, добавление товаров в корзину и оформление заказов);
2. Настройка JMeter: были созданы три различных тест-плана для моделирования нагрузки от 100, 500 и 1000 пользователей, и для каждого сценария был настроен соответствующий тайминг и количество виртуальных пользователей (threads) (каждый план включал настройку таймера задержки

между действиями, чтобы эмулировать естественные паузы между кликами пользователей);

3. Запуск тестов и сбор данных: после настройки сценариев запускались тесты, позволяя наблюдать за нагрузкой и производительностью приложения в реальном времени через JMeter (для мониторинга использовались различные метрики, такие как среднее время отклика на запросы, процент ошибок (HTTP-коды ошибок, тайм-ауты и т.д.) и количество обработанных запросов в минуту), и во время выполнения тестов мы отслеживали, как система справляется с увеличивающейся нагрузкой, чтобы выявить потенциальные узкие места и проблемные участки;
4. Анализ результатов: в результате каждого теста собирались и анализировались следующие данные:
   а) среднее время отклика – этот показатель позволил оценить, насколько быстро сервер обрабатывает запросы при различных уровнях нагрузки;
   б) количество ошибок – данные об ошибках помогли выявить критические моменты, когда сервер перестает справляться с запросами, что указывает на необходимость оптимизации;
   в) пиковая нагрузка – определялся момент, когда сервер достигал своих предельных возможностей.

После завершения всех тестов было проведено сравнение результатов и выявлены узкие места в текущей конфигурации системы. Например, при нагрузке в 1000 пользователей было замечено значительное увеличение времени отклика и повышение процента ошибок, что указывало на необходимость оптимизации производительности. Эти результаты послужили основой для дальнейших улучшений в инфраструктуре перед миграцией на AWS.

3. Анализ логов: для анализа логов с локального сервера был организован процесс сбора и обработки данных, чтобы понять частоту и продолжительность простоев. Мы начали с настройки системы централизованного логирования и детального аудита событий.

Шаги выполнения анализа логов:

1. Настройка системы логирования: для сбора логов было настроено логирование на локальном сервере, используя стандартные системные утилиты и конфигурационные файлы rsyslog и journald, что позволило автоматически собирать и сохранять логи всех критических событий системы и веб-сервера, включая запросы пользователей и записи о простоях;

2. Централизованное хранение логов: для удобства анализа мы настроили передачу логов в центральный сервер логирования, что позволило иметь единую точку доступа ко всем необходимым данным и выполнять их обработку более эффективно (логи с основного веб-сервера отправлялись через защищенное подключение на выделенный сервер с установленным Elastic Stack, а использование Filebeat помогло безопасно и автоматически пересылать логи, следя за новыми записями);

3. Анализ логов с использованием Kibana: с помощью программного обеспечения Kibana был выполнен предварительный анализ собранных данных, а именно созданы несколько визуализаций для мониторинга частоты сбоев и времени, в течение которого система была недоступна, на основе которых можно было увидеть временные интервалы и пиковые периоды, в которые происходили сбои;

4. Выявление закономерностей и проблемных точек: анализ логов по нескольким параметрам, таким как HTTP-коды ответов, сообщения об ошибках сервера, а также системные события, такие как перезагрузки и сбои, помог не только обнаружить сами факты сбоев, но и понять их первопричины (для этого была применена фильтрация по ключевым словам и временным меткам, выделяя все события, которые могли бы указывать на сбои);

5. Подготовка отчета: по итогам был составлен подробный отчет, в котором была представлена частота сбоев и их длительность за наблюдаемый период (отчет включал в себя графики, которые демонстрировали распределение простоев по времени и их продолжительность, а также критические ошибки, которые встречались наиболее часто, что помогло определить и предложить потенциальные способы устранения этих проблем).

Этот анализ позволил получить объективное представление о стабильности системы и точных временных интервалах, в которые происходили простои. Полученные данные легли в основу последующих решений по миграции инфраструктуры в AWS для улучшения доступности и надежности.

Данные о системе, полученные в результате тестирования исходной системы:

- доступность системы: 97.5% (среднее значение за неделю);
- время отклика: 450 мс при 500 пользователях, 700 мс при 1000 пользователях;
- частота простоев: 4 случая за неделю, каждый длительностью в среднем 30 минут;
- ошибки: около 3% ошибок в ответах сервера при нагрузке в 1000 пользователей.

Этап 2: Миграция инфраструктуры в AWS

Для развертывания интернет-магазина ООО «Тотал Продакшн» в AWS было проделано несколько шагов для применения и настройки созданной в рамках исследования отказоустойчивой и масштабируемой архитектуры.

Для начала мы организовали Multi-Region Deployment в два региона — eu-central-1 и ap-east-1. Основной регион расположен во Франкфурте и обрабатывает весь пользовательский трафик, а резервный в Гонконге готов к быстрому переключению в случае сбоя. Трафик маршрутизировался через AWS Global Accelerator, который распределяет запросы в зависимости от географического положения пользователя и состояния сервисов. Каждую минуту система проверяла состояние ресурсов, а переключение на резервный регион заняло 30 секунд благодаря настроенным Health Checks в Route 53.

Для обработки API-запросов, был настроен Amazon API Gateway который маршрутизировал AWS Lambda для выполнения серверной логики. Эти функции обрабатывали такие задачи, как поиск по каталогу товаров и произведение расчета стоимости аренды оборудования. Также Lambda-функции занимались

формированием корзины пользователя и обработкой платежей. Они были выбраны для реализации этого функционала из-за того, что они масштабировались автоматически, а время отклика оставалось стабильным даже при пиковых нагрузках.

Для более долгосрочных задач, (массовая обработка заказов), было выбрано использование сервиса AWS Fargate. Контейнеры запускались с конфигурацией 1 vCPU и 2GB RAM, а EventBridge и SQS управляли их запуском по мере поступления событий.

База данных интернет-магазина ООО «Тотал Продакшн» строилась на Amazon Aurora Global Database. Основной кластер находился в регионе eu-central-1, а резервный в ap-east-1. Это позволило синхронизировать данные между регионами менее чем за 1 секунду.

Для обеспечения высокой скорости доступа к каталогу товаров была добавлена DynamoDB, а чтобы снизить задержки, горячие данные кэшировались с помощью ElastiCache на основе Redis. Архивные данные и изображения товаров хранились в S3, где был настроен Intelligent Tiering, чтобы оптимизировать затраты.

Управление всей инфраструктурой было автоматизировано с помощью применения сервиса AWS CloudFormation. Каждый компонент системы, начиная от VPC и заканчивая настройками Lambda, описывался в шаблонах. Это позволило быстро вносить изменения и тестировать их перед применением. Для распределения нагрузки между серверами был настроен Elastic Load Balancer, а Auto Scaling Groups обеспечивали масштабирование EC2-инстансов при увеличении трафика.

Безопасность данных также была обеспечена. CloudFront, через который доставлялись статические файлы, не только ускорял их загрузку, но и защищал их с помощью WAF и Shield Advanced. Эти сервисы фильтровали вредоносные запросы и предотвращали DDoS-атаки. Пользовательская аутентификация реализовывалась через AWS Cognito, который поддерживал регистрацию и вход через социальные сети.

Этап 3: Тестирование системы после миграции в AWS

1. Повторное нагрузочное тестирование: Для проведения повторного нагрузочного тестирования после миграции мы следовалм тем же сценариям тестирования, которые использовались на начальном этапе. Основной целью было не только проверить стабильность приложения, но и оценить, насколько улучшились показатели времени отклика и количества ошибок в новой среде AWS.

Для этого этапа снова был использован Apache JMeter, так как он предоставлял возможность моделировать виртуальных пользователей и задавать определённые сценарии нагрузки. В JMeter мы предварительно создали скрипт, который эмулировал действия пользователей: авторизацию, просмотр страниц, добавление товаров в корзину и совершение покупки. Этот же скрипт использовался в исходных тестах, чтобы данные оставались сопоставимыми.

Был настроил план тестирования, включающий три группы пользователей (100, 500 и 1000 виртуальных пользователей), с постепенным наращиванием нагрузки и распределением запросов. Для каждой группы было указано:

- число потоков (пользователей) – 100, 500 и 1000;
- время разгона нагрузки, чтобы не допустить резких скачков;
- число циклов выполнения запросов;
- разнообразие сценариев (например, просмотр каталога, выполнение поиска, покупка).

Для проведения тестов была запущена каждая группа пользователей отдельно и мы внимательно следили за результатами, которые отображались в реальном времени. Мы включили такие параметры, как:

- среднее время отклика (Response Time), чтобы оценить, как быстро система реагирует на действия пользователей;
- процент ошибок (Error Rate), чтобы отслеживать некорректные ответы сервера.

Мы собрали результаты каждого теста в формате CSV для последующего анализа. Основное внимание уделялось:

1. Среднему и максимальному времени отклика;
2. Числу ошибок при увеличении нагрузки;
3. Плавности распределения нагрузки и изменения производительности системы под растущим числом пользователей.

После каждого теста мы делали экспорт отчетов в JMeter и выгружали данные на локальный сервер для последующего анализа. Также использовались отчёты JMeter, чтобы представить отклики и выявить возможные точки отказа или снижения производительности.

На основе повторного тестирования было выявлено значительное улучшение времени отклика при всех уровнях нагрузки, а также значительное снижение числа ошибок. Это подтвердило, что миграция в AWS и использование таких сервисов, как Elastic Load Balancer и Auto Scaling, эффективно справлялись с увеличением нагрузки.

Таким образом, повторное нагрузочное тестирование позволило сравнить показатели до и после миграции и убедиться, что внедрённые изменения привели к улучшению производительности и надежности веб-приложения.

2. Имитация высоких пиковых нагрузок: Для проведения теста автоматического масштабирования (Auto Scaling) мы спланировали имитацию пиковых нагрузок, чтобы проверить, как система справляется с резким увеличением числа пользователей. Цель заключалась в том, чтобы определить, насколько быстро AWS Auto Scaling сможет реагировать на растущие запросы и увеличивать количество ресурсов, чтобы поддерживать стабильную работу веб-приложения.

Перед началом тестов была проведена настройка Auto Scaling группы в AWS Management Console. Были установлены ключевые параметры:

– минимальное и максимальное количество EC2-инстансов: определены границы, чтобы система могла динамически увеличивать или уменьшать количество серверов;

– политики масштабирования: настроены политики масштабирования на основе загрузки процессора (CPU Utilization) и сетевого трафика (было задано пороговое значение нагрузки, при превышении которого система автоматически создаёт дополнительные инстансы).

Для имитации пиковых нагрузок нами снова был использован Apache JMeter. В сценарии мы запланировали постепенное увеличение числа пользователей:

1. Запуск с 500 пользователей для создания базовой нагрузки;

2. Резкое увеличение до 1000 и далее до 1500 пользователей, чтобы создать условия, схожие с пиковыми нагрузками (например, во время акций или рекламных кампаний).

При запуске теста мы наблюдали за несколькими метриками в режиме реального времени с использованием AWS CloudWatch. Основное внимание уделялось:

– загрузке процессора (CPU Utilization), показателю, на основе которого система принимает решение об увеличении или уменьшении числа инстансов;
– среднему времени отклика и количеству ошибок, показателю, отслеживаемые с помощью JMeter;
– активности Auto Scaling группы, а именно количеству добавленных или убранных инстансов и время выполнения этих операций.

Когда число пользователей достигло 1500, мы начали отслеживать, как система отреагировала на резкое увеличение нагрузки. При достижении порогового значения загрузки процессора (например, 70%) система сработала в соответствии с политиками Auto Scaling. В CloudWatch мы смогли увидеть:

– автоматическое добавление новых EC2-инстансов (количество инстансов увеличилось с 2 до 4);

– время реакции Auto Scaling составило около 2 минут, что позволило быстро восстановить стабильность и предотвратить снижение производительности.

Мы собрали данные о загрузке процессора и сетевом трафике в CloudWatch, а также результаты тестов в JMeter для анализа. Выяснилось, что добавление новых инстансов позволило поддерживать стабильное время отклика даже при нагрузке в 1500 пользователей. Система смогла автоматически справиться с увеличением нагрузки, сохранив доступность и производительность веб-приложения.

Проведённое тестирование подтвердило, что настроенные политики Auto Scaling в AWS эффективно реагируют на резкое увеличение нагрузки. Время отклика оставалось в допустимых пределах, а система показала стабильное поведение. Это подтвердило правильность выбранных настроек и эффективность автоматического масштабирования для обеспечения устойчивости приложения в условиях пиковых нагрузок.

3. Мониторинг доступности и анализа времени отклика: Для мониторинга доступности системы и анализа времени отклика были использованы возможности AWS CloudWatch. Основная задача заключалась в том, чтобы отслеживать стабильность работы веб-приложения после миграции в облако и фиксировать изменения в ключевых показателях доступности и производительности.

Первоначально мы создали несколько пользовательских дашбордов в AWS CloudWatch для удобного визуального наблюдения за состоянием системы. Были настроены метрики, специфичные для веб-приложения ООО «Тотал Продакшн». Для этого мы добавили специальные настройки мониторинга на уровне приложений, такие как HTTP-коды ответов (например, 200, 500) и количество активных соединений. Эти данные передавались в CloudWatch при помощи CloudWatch Agent.

Для мониторинга uptime системы мы использовали функцию CloudWatch Alarms, которая позволяла отслеживать состояние инстансов и доступность

приложения. В случае возникновения проблемы (при снижении уровня доступности ниже 99.99%) система отправляла уведомления через Amazon SNS (Simple Notification Service). Эти уведомления позволили нам быстро реагировать на потенциальные сбои.

Для анализа времени отклика был создан и настроен Amazon CloudWatch Metrics для отслеживания задержек при ответах на HTTP-запросы. Это было реализовано путём установки таймеров в приложении, которые фиксировали время обработки каждого запроса на сервере. Каждый раз, когда запрос завершался, данные отправлялись в CloudWatch, где они агрегировались и анализировались.

Мы регулярно просматривали собранные данные в дашбордах CloudWatch, чтобы выявить возможные отклонения в метриках. При этом мы обращали внимание на то, что:

- графики uptime показывали стабильность работы системы и время безотказной работы в течение каждой недели;
- графики времени отклика, которые отображали изменение задержек при ответах на запросы, особенно в условиях повышенной нагрузки (мы настроили агрегирование данных по средним значениям и перцентилям, чтобы выявить максимальные отклонения и пики нагрузки);
- логи ошибок: при помощи CloudWatch Logs мы собирали данные о системных ошибках и HTTP-кодах для быстрого выявления причин снижения доступности или повышения времени отклика.

На основе собранных данных было проанализировано, как система справлялась с изменением нагрузки и реагировала на различные сценарии. CloudWatch позволял нам создавать визуализации и графики, на которых мы могли видеть закономерности и тренды, такие как:

- время отклика при увеличении числа пользователей;
- периоды с повышенной или пониженной доступностью;
- корреляции между пиковыми нагрузками и ростом HTTP-ошибок.

На основе анализа собранных данных мы смогли скорректировать политики Auto Scaling и настройки Elastic Load Balancer, чтобы оптимизировать распределение нагрузки и минимизировать время отклика. Также были настроены более точные алерты для автоматического оповещения о снижении доступности или резком увеличении времени отклика, что позволило оперативно реагировать на любые проблемы.

Таким образом, мониторинг через AWS CloudWatch обеспечил непрерывное наблюдение за системой и предоставил точные данные для анализа производительности и доступности. Это позволило своевременно выявлять и устранять проблемы, улучшать стабильность и эффективность работы веб-приложения.

Данные о системе, полученные в результате тестирования системы после миграции в AWS:

- доступность системы: 99.99% (среднее значение за неделю);
- время отклика: 150 мс при 500 пользователях, 220 мс при 1000 пользователях;
- частота простоев: 1 случай за неделю, длительность 5 минут;
- ошибки: менее 0.5% при 1000 пользователях;
- время реакции на изменение нагрузки – 2 минуты (инстансы добавились и удалились автоматически).

Этап 4: Сравнительный анализ и апробация гипотезы.

Миграция инфраструктуры веб-приложения в AWS позволила значительно повысить его надежность и производительность, что отображено в таблице 3:

Таблица 3 – Сравнение показателей инфраструктуры веб-приложения до и после миграции в облачные сервисы AWS

Параметр	До миграции	После миграции (AWS)

метр		
Доступность системы	97.5%	99.99%
Время отклика (500 пользователей)	450 мс	150 мс
Время от	700 мс	220 мс

клика (10000 пользователей)		

Продолжение таблицы 3

Частота простоев	4 случая/неделя	1 случай/неделя
Средняя длительность простоев	30 минут	5 минут
Процент ошибок (1000 пользователей)	3%	< 0.5%
Время реакции на изменение нагрузки	-	2 минуты

Уровень доступности вырос с 97.5% до 99.99% (рисунок 7):

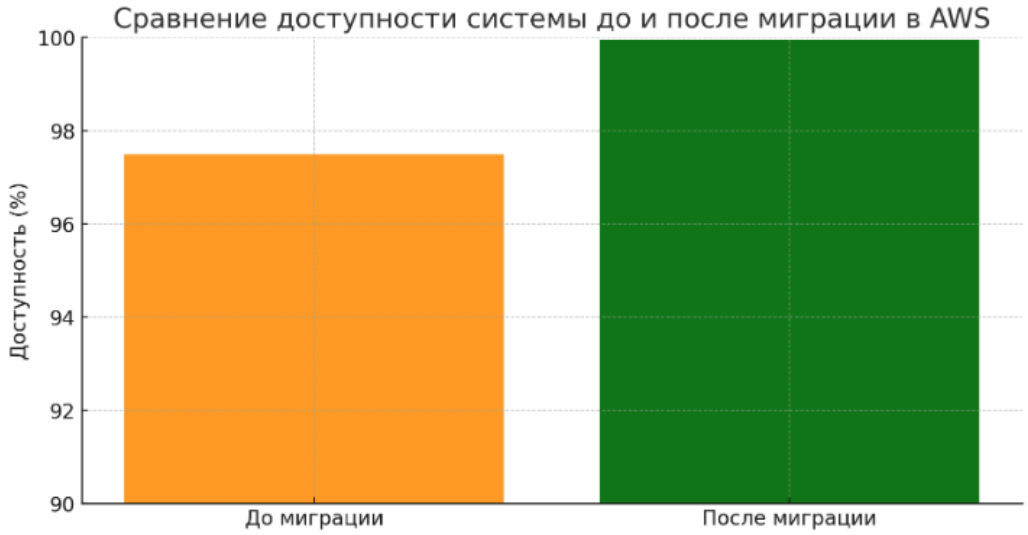

Рисунок 7 – Диаграмма сравнения доступности системы до и после миграции в AWS

Время отклика при увеличении нагрузки заметно сократилось: для 500 пользователей с 450 мс до 150 мс, а для 1000 пользователей — с 700 мс до 220 мс (рисунок 8):

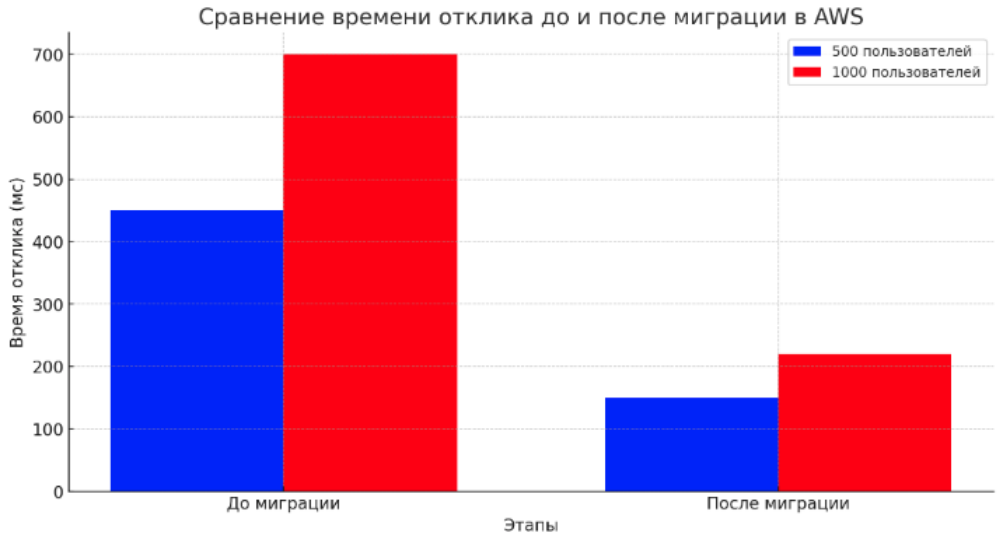

Рисунок 8 – Диаграмма сравнения времени отклика системы до и после миграции в AWS

Частота и продолжительность простоев также уменьшились, а процент ошибок снизился с 3% до менее 0.5%. Кроме того, система приобрела способность адаптироваться к изменениям нагрузки всего за 2 минуты (рисунок 9):

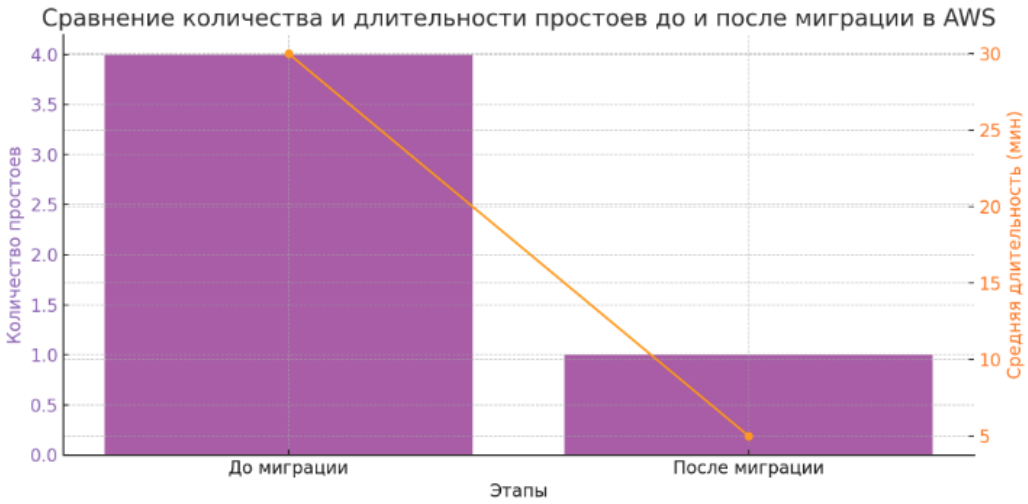

Рисунок 9 – Диаграмма сравнения количества и длительности простоев системы до и после миграции в AWS

Эти результаты подтверждают, что использование облачной инфраструктуры позволило создать более стабильную и высокопроизводительную систему.

Выводы по третьей главе

Анализ структуры и процессов ООО «Тотал Продакшн» показал, что в условиях планируемого запуска интернет-магазина и выхода на международный рынок значимость IT-инфраструктуры существуюшего веб-приложения существенно возрастает, превращаясь в стратегический ресурс для развития.

Был разработан план миграции и оптимизации веб-приложения компании. Это легло в основу проведенного эксперимента, в ходе которого была разработана и протестирована новая архитектура веб-приложения ООО «Тотал Продакшн» с учетом требований к масштабируемости и высокой доступности. Миграция инфраструктуры веб-приложения в AWS позволила значительно повысить его надежность и производительность.

Для оценки результатов эксперимента был использован метод сравнительного анализа, а также статистическая обработка данных для определения значимости изменений. В данном эксперименте основное внимание уделялось ключевым метрикам, таким как доступность системы, время отклика, количество и длительность простоев, возможность приложения к масштабированию также было учтено.

Результаты миграции инфраструктуры веб-приложения ООО «Тотал Продакшн» в AWS наглядно подтверждают гипотезу о том, что если для проектирования системы интернет-магазина будут применены облачные технологии, то система сможет сможет эффективно масштабироваться в ответ на изменяющийся интернет-трафик, обеспечивая высокую доступность и отказоустойчивость, минимизируя простои и перерывы в обслуживании, а также сохранять оптимальную производительность при изменении нагрузки. Использование облачных технологий помогает эффективно решать ключевые бизнес-задачи и оптимизировать работу систем. Такие результаты говорят о том, что использование AWS не просто улучшает технические параметры системы,

но и предоставляет бизнесу новые возможности для роста и повышения эффективности.

Таким образом, эксперимент с миграцией веб-приложения ООО «Тотал Продакшн» в AWS подтверждает, что облачная инфраструктура способна не только решать технические проблемы, но и помогать бизнесу адаптироваться к динамичным рыночным условиям, увеличивать операционную эффективность и создавать конкурентные преимущества.

Заключение

Данная магистерская работа была посвящена исследованию технологии интеграции Amazon Web Services (AWS) для проектирования и оптимизации веб-приложений для обеспечения высокой доступности и масштабируемости. В рамках исследования была рассмотрена реальная задача ООО «Тотал Продакшн» по модернизации инфраструктуры интернет-магазина в целях его выхода на международный рынок. Выполненные задачи и сделанные выводы подтвердили актуальность, значимость и практическую ценность работы.

Итоговые выводы:

1. Проведенный анализ показал, что облачные технологии превосходят локальные серверные инфраструктуры по гибкости, экономической эффективности, масштабируемости и отказоустойчивости. Использование облачных решений позволяет компаниям минимизировать капитальные затраты на оборудование и снизить риски, связанные с отказами. Для интернет-магазинов, нацеленных на международный рынок, облачные платформы являются оптимальным выбором, так как они обеспечивают быструю адаптацию к изменяющимся условиям и высокую доступность.
2. Сравнение функциональности облачных провайдеров позволило установить, что AWS предоставляет наиболее универсальный и экономически эффективный набор инструментов для масштабирования и обеспечения отказоустойчивости веб-приложений. Платформа AWS отличается гибкой системой оплаты, высокой скоростью обработки данных и развитой экосистемой, что делает ее подходящей для интернет-магазинов, ориентированных на международную аудиторию.
3. Разработанна уникальная гибридная архитектура на основе AWS, построенная на основе серверлес-решений и управляемых облачных сервисов. Такая архитектура доказала свою эффективность в обеспечении высокой производительности, минимизации простоев и быстрого отклика

на изменения нагрузки. Экономическая эффективность достигается за счет оптимизации расходов на инфраструктуру.

4. Результаты тестирования подтвердили, что переход на облачную платформу AWS существенно улучшил ключевые показатели интернет-магазина ООО «Тотал Продакшн» в области обеспечения высокой доступности и масштабируемости:
   - доступность системы увеличилась с 97.5% до 99.99%;
   - время отклика сократилось с 450 мс до 150 мс при нагрузке в 500 пользователей;
   - частота простоев снизилась с 4 случаев в неделю до 1 случая;
   - ошибки при высокой нагрузке уменьшились с 3% до менее 0.5%.

Полученные в работе результаты могут быть использованы не только ООО «Тотал Продакшн», но и другими компаниями, планирующими масштабирование своих веб-приложений с учетом международных стандартов и требований. Разработанная архитектура является универсальным инструментом, который рекомендован за основу для построения высокодоступной и масштабируемой архитектуры для веб-приложения. Немаловажным является и тот факт, что предложенное решение является адаптируемым под специфику различных проектов электронной коммерции.

Настоящее исследование открывает перспективы для изучения более сложных сценариев интеграции облачных технологий, таких как использование искусственного интеллекта для предиктивного масштабирования, внедрение автоматизированного мониторинга безопасности и разработка решений для интеграции мультиоблачных платформ. Также целесообразно исследовать возможности использования облачных технологий в других отраслях, где требуется высокая доступность и масштабируемость.

Таким образом, результаты этого исследования позволяют не только решить текущие задачи компании, но и расширяют знания в области оптимизации и миграции веб-приложений в облачные инфраструктуры.

По окончанию эксперимента было подтверждено, что миграция веб-приложения в AWS привела к значительному улучшению по всем ключевым критериям:

- повышение доступности системы на 2.49%.
- существенное снижение времени отклика под высокой нагрузкой.
- уменьшение частоты и длительности простоев.
- успешная реализация автоматического масштабирования для улучшения производительности.

На основе данных, полученных в ходе эксперимента, гипотеза о том, что если для проектирования системы интернет-магазина будут применены облачные технологии, то система сможет эффективно масштабироваться в ответ на изменяющийся интернет-трафик, обеспечивая высокую доступность и отказоустойчивость, минимизируя простои и перерывы в обслуживании, а также сохранять оптимальную производительность при изменении нагрузки подтверждена.

Список используемых источников

1. Александровский С.В., Бутрюмова Н.Н. Исследование рынка облачных сервисов в Нижегородском регионе // Региональная экономика: теория и практика. 2016. №5 (428). URL: https://cyberleninka.ru/article/n/issledovanie-rynka-oblachnyh-servisov-v-nizhegorodskom-regione (дата обращения: 28.09.2024).
2. Белоножко П.П., Белоус В. В., Куцевич Н. А., Храмов Д. А. Свободные облачные аппаратно-программные платформы. Аналитический обзор // Вестник евразийской науки. 2016. №6 (37). URL: https://cyberleninka.ru/article/n/svobodnye-oblachnye-apparatno-programmnye-platformy-analiticheskiy-obzor (дата обращения: 13.10.2024).
3. Богданов А. В., Е М. Н. Сравнение нескольких платформ облачных вычислений // Вестник СПбГУ. Серия 10. Прикладная математика. Информатика. Процессы управления. 2013. №2. URL: https://cyberleninka.ru/article/n/sravnenie-neskolkih-platform-oblachnyh-vychisleniy (дата обращения: 15.10.2024).
4. Борисова А. А., Борисенко О. Д. ИССЛЕДОВАНИЕ МЕТОДОВ ПОСТРОЕНИЯ ОБЛАЧНЫХ ПЛАТФОРМЕННЫХ СЕРВИСОВ И РЕАЛИЗАЦИЙ СТАНДАРТА TOSCA // Труды ИСП РАН. 2022. №5. URL: https://cyberleninka.ru/article/n/issledovanie-metodov-postroeniya-oblachnyh-platformennyh-servisov-i-realizatsiy-standarta-tosca (дата обращения: 20.10.2024).
5. Булиньш З. А. Миграция баз данных MySQL в облачную среду Google Cloud sql и возможности её применения в сфере образования // ОТО. 2012. №3. URL: https://cyberleninka.ru/article/n/migratsiya-baz-dannyh-mysql-v-oblachnuyu-sredu-google-cloud-sql-i-vozmozhnosti-eyo-primeneniya-v-sfere-obrazovaniya (дата обращения: 03.10.2024).

6. Васяткин М. А., Белоус К. В. - Облачное хранилище данных // StudNet. 2020. №10. URL: https://cyberleninka.ru/article/n/oblachnoe-hranilische-dannyh-1 (дата обращения: 17.10.2024).
7. Воронцов А. И., Бусенков А. А., Куприков О. Д. АНАЛИЗ РАЗЛИЧНЫХ ИНСТРУМЕНТОВ УПРАВЛЕНИЯ И МОНИТОРИНГА ОБЛАЧНОЙ ИНФРАСТРУКТУРОЙ // Экономика и качество систем связи. 2022. №2 (24). URL: https://cyberleninka.ru/article/n/analiz-razlichnyh-instrumentov-upravleniya-i-monitoringa-oblachnoy-infrastrukturoy (дата обращения: 14.09.2024).
8. Гордина А. Т., Забродин А. В. Особенности технологий бессерверных вычислений // Интеллектуальные технологии на транспорте. 2022. №1 (29). URL: https://cyberleninka.ru/article/n/osobennosti-tehnologiy-besservernyh-vychisleniy (дата обращения: 19.09.2024).
9. Городничев М. Г., Кочупалов А. Е. Исследование методов межпроцессного взаимодействия в информационной системе с горизонтальным взаимодействием // Вестник евразийской науки. 2018. №4. URL: https://cyberleninka.ru/article/n/issledovanie-metodov-mezhprotsessnogo-vzaimodeystviya-v-informatsionnoy-sisteme-s-gorizontalnym-vzaimodeystviem (дата обращения: 11.09.2024).
10. Горюнова М. П. Архитектурные стили в разработке web-приложений и область их применения // Проблемы Науки. 2017. №12 (94). URL: https://cyberleninka.ru/article/n/arhitekturnye-stili-v-razrabotke-web-prilozheniy-i-oblast-ih-primeneniya (дата обращения: 30.10.2024).
11. Григорьев В. Р., Кузнецов В. С. Проблемы выявления уязвимостей в модели облачных вычислений // Спецтехника и связь. 2012. №4. URL: https://cyberleninka.ru/article/n/problemy-vyyavleniya-uyazvimostey-v-modeli-oblachnyh-vychisleniy (дата обращения: 04.09.2024).
12. Гусев А. В. Перспективы облачных вычислений и информатизация учреждений здравоохранения // Врач и информационные технологии. 2011. №2. URL: https://cyberleninka.ru/article/n/perspektivy-oblachnyh-

vychisleniy-i-informatizatsiya-uchrezhdeniy-zdravoohraneniya (дата обращения: 17.09.2024).

13. Исхакова Л. М. Использование облачных сервисов ArcGIS Online и Microsoft Azure для анализа деятельности сети АЗС // Перспективы развития информационных технологий. 2014. №19. URL: https://cyberleninka.ru/article/n/ispolzovanie-oblachnyh-servisov-arcgis-online-i-microsoft-azure-dlya-analiza-deyatelnosti-seti-azs (дата обращения: 08.09.2024).
14. Казакова У. А. ПРИМЕНЕНИЕ ТЕХНОЛОГИИ CLOUD COMPUTING // E-Scio. 2021. №3 (54). URL: https://cyberleninka.ru/article/n/primenenie-tehnologii-cloud-computing (дата обращения: 02.10.2024).
15. Клементьев С. А. ПРИМЕНЕНИЕ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ // Экономика и социум. 2016. №6-3 (25). URL: https://cyberleninka.ru/article/n/primenenie-oblachnyh-vychisleniy (дата обращения: 18.10.2024).
16. Копылов Д. В. АНАЛИЗ И ПОСТРОЕНИЕ МОДЕЛИ ОТКАЗОУСТОЙЧИВОЙ ИНФРАСТРУКТУРЫ ДЛЯ РАЗМЕЩЕНИЯ ECOMMERCE САЙТОВ MAGENTO НА ОСНОВЕ ОБЛАЧНЫХ РЕШЕНИЙ // Столыпинский вестник. 2022. №9. URL: https://cyberleninka.ru/article/n/analiz-i-postroenie-modeli-otkazoustoychivoy-infrastruktury-dlya-razmescheniya-ecommerce-saytov-magento-na-osnove-oblachnyh (дата обращения: 24.10.2024).
17. Ойбек З. СПОСОБЫ ЭФФЕКТИВНОГО ИСПОЛЬЗОВАНИЯ «ОБЛАЧНЫХ ТЕХНОЛОГИЙ» В СИСТЕМЕ ОБРАЗОВАНИЯ // Вестник науки и образования. 2020. №22-3 (100). URL: https://cyberleninka.ru/article/n/sposoby-effektivnogo-ispolzovaniya-oblachnyh-tehnologiy-v-sisteme-obrazovaniya (дата обращения: 28.09.2024).
18. Костюков А. А. Проблемы управления нагрузкой сетевых узлов // APRIORI. Серия: Естественные и технические науки. 2016. №1. URL:

https://cyberleninka.ru/article/n/problemy-upravleniya-nagruzkoy-setevyh-uzlov (дата обращения: 16.10.2024).

19. Куваев М. Ю., Антимонов О. В. Современные тенденции развития веб-разработки // StudNet. 2020. №9. URL: https://cyberleninka.ru/article/n/sovremennye-tendentsii-razvitiya-veb-razrabotki (дата обращения: 31.10.2024).
20. Кузнецов А. Ф., Шабанов А. А. Преимущества и недостатки использования облачных технологий // Огарёв-Online. 2015. №15 (56). URL: https://cyberleninka.ru/article/n/preimuschestva-i-nedostatki-ispolzovaniya-oblachnyh-tehnologiy (дата обращения: 25.09.2024).
21. Лазарев Н. А., Борисенко О. Д. ПОСТРОЕНИЕ ТРЕБОВАНИЙ И АРХИТЕКТУРЫ ОБЛАЧНОГО ОРКЕСТРАТОРА ПЛАТФОРМЕННЫХ СЕРВИСОВ // Труды ИСП РАН. 2022. №4. URL: https://cyberleninka.ru/article/n/postroenie-trebovaniy-i-arhitektury-oblachnogo-orkestratora-platformennyh-servisov (дата обращения: 06.10.2024).
22. Лазеева В. И., Токмачева М. В. Автоматизация бизнес-процессов предприятий малого бизнеса на платформе публичного облака // Актуальные проблемы авиации и космонавтики. 2016. №12. URL: https://cyberleninka.ru/article/n/avtomatizatsiya-biznes-protsessov-predpriyatiy-malogo-biznesa-na-platforme-publichnogo-oblaka (дата обращения: 23.09.2024).
23. Левина А. И., Кубарский А. В. Преимущества использования SaaS программного обеспечения в сравнении с on-premises программным обеспечением // Научный вестник ЮИМ. 2018. №4. URL: https://cyberleninka.ru/article/n/preimuschestva-ispolzovaniya-saas-programmnogo-obespecheniya-v-sravnenii-s-on-premises-programmnym-obespecheniem (дата обращения: 28.10.2024).
24. Лобанов А. А. Облачные вычисления как развитие информационного сервиса // ПНиО. 2014. №2 (8). URL:

https://cyberleninka.ru/article/n/oblachnye-vychisleniya-kak-razvitie-informatsionnogo-servisa (дата обращения: 30.09.2024).

25. Логиновский О. В., Шестаков А. Л., Шинкарев А. А. Построение современных корпоративных информационных систем // УБС. 2019. №81. URL: https://cyberleninka.ru/article/n/postroenie-sovremennyh-korporativnyh-informatsionnyh-sistem (дата обращения: 02.10.2024).
26. Макосий А. И., Макосий Р. Современная облачная инфраструктура: бессерверные вычисления // Вестник ХГУ им. Н. Ф. Катанова. 2019. №28. URL: https://cyberleninka.ru/article/n/sovremennaya-oblachnaya-infrastruktura-besservernye-vychisleniya (дата обращения: 01.09.2024).
27. Мархакшинов А. Л., Тонхоноева А. А., Урмакшинова Е. Р. Разработка бессерверных мобильных приложений // Вестник НГУ. Серия: Информационные технологии. 2019. №4. URL: https://cyberleninka.ru/article/n/razrabotka-besservernyh-mobilnyh-prilozheniy (дата обращения: 24.10.2024).
28. Мурзин Ф. А., Батура Т. В., Семич Д. Ф. Облачные технологии: основные модели, приложения, концепции и тенденции развития // Программные продукты и системы. 2014. №3 (107). URL: https://cyberleninka.ru/article/n/oblachnye-tehnologii-osnovnye-modeli-prilozheniya-kontseptsii-i-tendentsii-razvitiya-1 (дата обращения: 31.10.2024).
29. Ольхов Д. А. АНАЛИЗ ПОДХОДОВ К УПРАВЛЕНИЮ ПРИКЛАДНЫМ ПРОГРАММНЫМ ИНТЕРФЕЙСОМ МИКРОСЕРВИСОВ В ОБЛАЧНОЙ СРЕДЕ // Символ науки. 2021. №3. URL: https://cyberleninka.ru/article/n/analiz-podhodov-k-upravleniyu-prikladnym-programmnym-interfeysom-mikroservisov-v-oblachnoy-srede (дата обращения: 19.10.2024).
30. Пантелеев Н. Н., Панов С. С., Кудояр В. Р. АНАЛИЗ И ВЫБОР СТРУКТУРЫ ОЦЕНКИ РИСКОВ ДЛЯ ОБЛАЧНЫХ ПРИЛОЖЕНИЙ // E-Scio. 2022. №4 (67). URL: https://cyberleninka.ru/article/n/analiz-i-vybor-

struktury-otsenki-riskov-dlya-oblachnyh-prilozheniy (дата обращения: 20.09.2024).

31. Andreev R. A., Krotova E. L. Analysis of cloud computing segment in Russia // МНИЖ. 2016. №4-2 (46). URL: https://cyberleninka.ru/article/n/analysis-of-cloud-computing-segment-in-russia (дата обращения: 05.06.2024).
32. Kabarukhin A. P. MODERN CLOUD INFRASTRUCTURE: SERVERLESS COMPUTING // Наука, техника и образование. 2022. №2 (85). URL: https://cyberleninka.ru/article/n/modern-cloud-infrastructure-serverless-computing (дата обращения: 25.06.2024).
33. Petrenko S. SELF-HEALING CLOUD COMPUTING // Вопросы кибербезопасности. 2021. №1 (41). URL: https://cyberleninka.ru/article/n/self-healing-cloud-computing (дата обращения: 14.06.2024).
34. Seyidova I., Hashimov O. CLOUD COMPUTİNG: A REVİEW OF THE AVAİLABLE PLATFORMS // Sciences of Europe. 2022. №107. URL: https://cyberleninka.ru/article/n/cloud-computing-a-review-of-the-available-platforms (дата обращения: 20.05.2024).
35. Volkov A. O. EVALUATION OF CLOUD COMPUTING CLUSTER PERFORMANCE // T-Comm. 2020. №12. URL: https://cyberleninka.ru/article/n/evaluation-of-cloud-computing-cluster-performance (дата обращения: 17.06.2024).